Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Monitorización de sensores con arduino utilizando el protocolo MQTT

TITULACIÓN: Ingeniería de Sistemas de Telecomunicaciones

AUTORES: Mario Calleja Collado, Joan Guasch Llobera

DIRECTOR: David Matanzas

SUPERVISOR: Ramon Casanella

DATA: 02/05/2019

Título: Monitorización de sensores con arduino utilizando el protocolo MQTT

Autores: Mario Calleja Collado, Joan Guasch Llobera

Director: David Matanzas

Supervisor: Ramon Casanella

Fecha: 02/05/2019

Resumen

Vivimos en una época donde la tecnología es muy importante y se está implementando en todos los ámbitos (trabajo, servicios públicos, viviendas, etc). Es por ello que están surgiendo nuevos servicios y nuevos métodos de implementación basado en IoT. Estos servicios y protocolos necesitan ser estudiados antes y durante la implementación para poder ser actualizados y adaptados correctamente a las necesidades de los usuarios ya sean clientes particulares o empresas.

Por este motivo, en este proyecto se documenta una solución que tiene como objetivo la implementación del protocolo MQTT para la gestión de componentes, como por ejemplo leds y sensores (humedad, temperatura, gas, presencia, sonido, etc). Con estos componentes se puede dar servicio a ciertas necesidades básicas que tienen los usuarios. Por lo contrario, si se trata de dispositivos que requieren un gran flujo de datos constantes y cierta velocidad de transmisión, como podrían ser cámaras de video vigilancia, se tendría que usar otros tipos de protocolos, como podrían ser RTSP, HTTP, entre otros.

En vista de que este proyecto se basa en la utilización de sensores simples, se utiliza el protocolo MQTT, dónde la principal característica que tiene es que la comunicación al realizarse M2M, es decir, que hay un maestro y uno o varios esclavos, requiere poco ancho de banda y poco consumo, por eso es ideal para sensores que solo transmiten información cada cierto tiempo. Otra característica es que es fácil delimitar en qué espacios se tienen distribuidos los sensores puesto que se usa el método de publicar y suscribirse a topics.

Para concluir, se ha realizado un montaje a modo de esquema de lo que se podría llegar a implementar en una situación real, con un led y un sensor de humedad y temperatura. Donde la comunicación se realiza mediante el protocolo MQTT y un bróker en la nube, donde se puede suscribir a los topics de temperatura, humedad y encender/apagar led.

Title: Monitoring of sensors with arduino using the MQTT protocol

Author: Mario Calleja Collado, Joan Guasch Llobera

Director: David Matanzas

Supervisor: Ramon Casanella

Date: 02/05/2019

Overview

We live in an era in which technology is very important and it is being implemented in all aspects (work, public services, housing, etc.). It is because of this that there are emerging new services and new methods of implementation based on IoT. These services and protocols need to be studied before and during the implementation to be able to be updated and adapted properly to the needs of the users, whether they are particular clients or corporations.

For this reason, in this project a solution has been documented, which has as its target the implementation of a protocol, MQTT, for the management of components, like LEDs and sensors (humidity, temperature, gas, presence, sound, etc.). With these components, service can be given to certain basic needs that consumers may have.

On the other side, if it is about devices that require a big data flow and a certain transmission speed, as it could be with surveillance cameras, other protocols such as RTSP and HTTP should be used among others.

Taking a look on the fact that this project is based on the use of simple sensors, the MQTT protocol is used, where the main characteristic is that communication when M2M is performed, that is to say, there is a master and one or more slaves, requires little bandwidth and low consumption, so it is ideal for sensors that only transmit information from time to time. Another feature is that it is easy to define in which spaces the sensors are distributed since the method of publishing and subscribing to topics is used.

To conclude, an assembly has been made as a scheme of what could be implemented in a real situation, with an LED and both a humidity and temperature sensor, where the communication is established through the MQTT protocol and a broker in the cloud, where you can subscribe to the topics of temperature, humidity and turn on/off the LEDs.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

API: Interfaz de programación de aplicaciones
CoAP: Protocolo de aplicación restringida
CPU: Unidad central de procesamiento
DIY: Do it yourself
HTTP: Protocolo de transferencia de hipertexto
IMU: Unidad de medición inercial
IoT: Internet de las cosas
IP: Internet protocolo
M2M: Machine-to-Machine
MQTT: Message queue server telemetry transport
NFC: Comunicación de campo cercano
OSI: Interconexión de sistemas abiertos
QoS: Calidad del Servicio
RAM: Memoria de acceso aleatorio
RFID: Identificación por radiofrecuencia
SSL: Secure sockets layer
TCP: Protocolo de control de transmisión
XMPP: Protocolo extensible de mensajería y presencia

INTRODUCCIÓN

El siguiente trabajo final de grado se basa en hablar sobre el protocolo de comunicación MQTT que se usa en el ámbito de las IoT, un entorno que no para de crecer y como consecuencia surgen nuevos protocolos para la comunicación entre los sensores y el servidor.

Este crecimiento se puede obeservar en la siguiente gráfica, realizada con la herramiento Google Trends, que permite analizar las búsquedas realizadas en google durante un período de tiempo y un espacio geográfico concreto. Concretamente, se estudia el período 2014-actualidad y las búsquedas realizadas en todo el mundo.

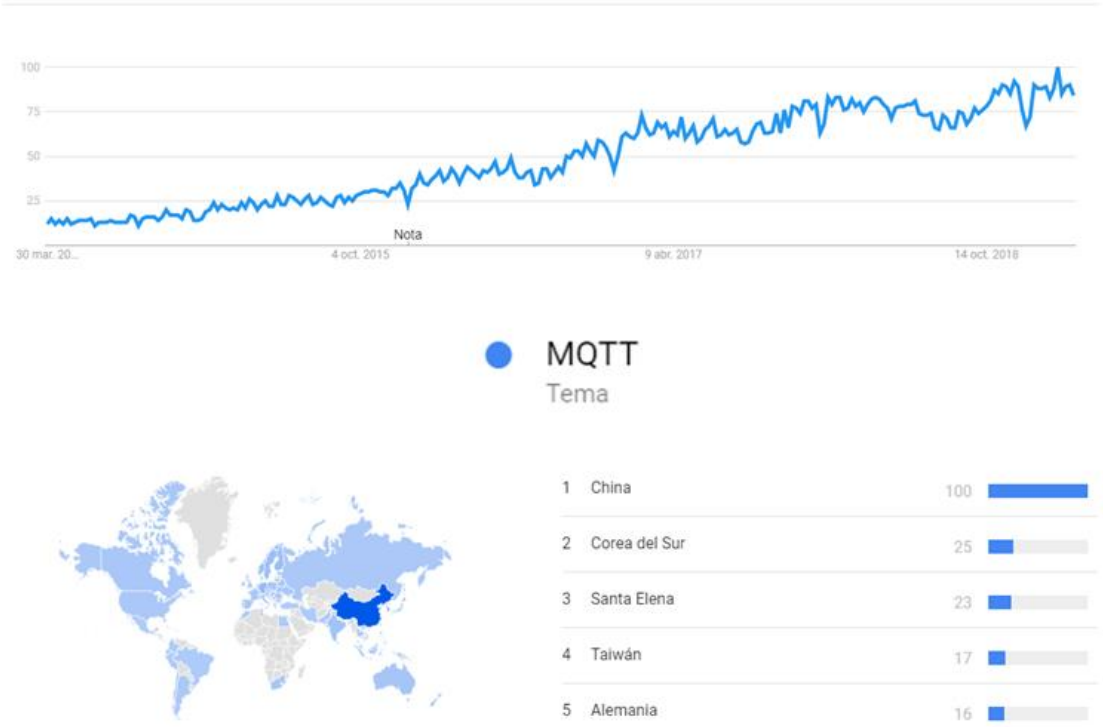

Fig. 0.1 Resultados de la búsqueda MQTT de Google Trends

En la gráfica se observa que desde el año 2014 se han ido incrementando las búsquedas sobre el protocolo de comunicación MQTT, por tanto, se aprecia un incremento significativo de la utilización de este protocolo. Otro dato importante que se visualiza en el mapa mundial es el número de búsquedas que se realiza en cada país, donde es imporante destacar que China se trata del país más puntero desarollando aplicaciones IoT con este protocolo de comunicación.

Por esta razón se ha decidido realizar el estudio del protocolo que, como se verá en el proyecto, es el mejor para gestionar componentes que necesitan poco

ancho de banda, como podrían ser leds o sensores (humedad, temperatura, gas, presencia, sonido, etc).

Con la finalidad de entender la situación en que se ambienta el trabajo, se explicará en el capítulo dos el entorno en que se encuentra actualmente las IoT y los protocolos de comunicación más usados.

Para demostrar los motivos del porqué es mejor usar el protocolo de comunicación MQTT respecto a otros y usando el estudio comparativo de protocolos, se ha realizado un estudio con las ventajas e inconvenientes que lo hacen el más óptimo para gestionar el tráfico de datos generado por los sensores. Por otra parte, se explican los tipos de sensores que son óptimos para el protocolo utilizado y dónde van conectados, que en el caso de este proyecto se trata de una o varias placas arduino, donde se explica el motivo por el que se decide escoger este tipo de placa. Todo esto se ve reflejado en el capítulo tres del proyecto.

El proyecto tiene una parte práctica, una demostración básica del potencial que tiene usar el protocolo de comunicación MQTT respecto los demás. Esta parte se explicará mediante los procesos que se han ido realizando para llegar al objetivo deseado en el capítulo cuatro.

Para terminar con el proyecto, durante el capítulo cinco, se llevarán a cabo las conclusiones y valoraciones globales del proyecto. Junto con la valoración de los resultados obtenidos durante el proyecto.

CAPÍTULO 1. OBJETIVOS Y MOTIVACIONES

En el primer capítulo del proyecto se pondrá en contexto todos los objetivos que tiene el proyecto referente a la utilización del protocolo de comunicación MQTT y la utilización de los sensores. Junto con las motivaciones que ha hecho movernos para que desarrollemos este proyecto.

1.1. Objetivos del proyecto

Este proyecto tiene como objetivo estudiar con profundidad el protocolo de comunicación MQTT ya que es el más usado para comunicaciones M2M de IoT entre sensores que no requieren un gran ancho de banda, el broker y servidor.
Primero se pondrá en contexto las IoT y se explicarán los conceptos que lo definen como tal. Se tendrá en cuenta aspectos como la seguridad y privacidad, la legalidad y la interoperabilidad. Así pues, se explicará en que ámbitos son más comunes las IoT o en que ámbitos emergen con una mayor fuerza o una mayor proyección de futuro.
Se explicará en qué consiste este protocolo, cuándo se debe usar, qué arquitectura tiene, cómo de seguro es, qué ventajas tiene respecto a otros protocolos similares como CoAP, HTTP, etc.
El proyecto constará de una segunda parte práctica dónde se implementará el protocolo MQTT en un pequeño entorno de prueba que constará de un sensor de temperatura/humedad y un LED conectados a una Arduino UNO que será la encargada de enviar la información del sensor y el LED al broker MQTT (Cloud MQTT). A este broker se podrá acceder mediante un PC o un móvil que harán de clientes y se suscribirán a los topics de temperatura, humedad y led, de tal modo que se podrá controlar el encendido o apagado del led así como obtener información de su estado (encendido o apagado), o se informará del nivel de temperatura o humedad que recibe el sensor de temperatura y humedad.
Para realizar lo dicho anteriormente, habrá que programar la Arduino UNO mediante el lenguaje de programación C y se explicará en qué entorno se ha programado, las librerías que se han importado, como se definen variables para los pines de Arduino, cómo se realiza la petición al servidor MQTT y cómo funciona la dinámica de publicar y suscribirse a topics.
En resumen, la finalidad del proyecto será estudiar la comunicación entre los sensores, el broker y los clientes así como su funcionamiento y coste de implementación.

1.2. Motivaciones

La principal motivación que tenemos para realizar el proyecto es que nos interesa mucho poder realizar una domotización sencilla en nuestras respectivas viviendas. Como por ejemplo, poder instalar sensores de movimiento para el encendido y apagado de luces y/o sensores de temperatura distribuidos en

puntos clave de la casa. De esta manera se puede ver la diferencia exacta que hay de temperatura.

La idea es poder ver qué protocolo de comunicación es más óptimo para el diseño y monitorización de la implementación de los sensores que se quieren usar en el diseño basado en la idea de las IoT. De esta manera, se podrá observar que el protocolo de comunicación más óptimo para la utilización de los sensores que se quieren usar es el de MQTT.

Otra de las motivaciones que se tiene para realizar el proyecto es poder entender y profundizar más en la idea de las IoT, ya que en la actualidad la mayoría de proyectos se basan en este idea, incluso las grandes ciudades como Barcelona, en sus proyectos tecnológicos intentan seguir esta idea. Por este motivo y gracias a la realización de este proyecto que estudia las bases e implementaciones de la idea de las IoT, se puede entender más esta idea.

Para finalizar, se tiene que tener presente que la idea del “Internet of Things” es muy compleja y en este proyecto solo se realiza un estudio de una parte muy concreta, que es la necesaria para realizar este proyecto

CAPÍTULO 2. INTERNET OF THINGS

2.1. Aspectos relevantes

Este concepto de IoT se refiere a que los objetos cotidianos están interconectados entre ellos mediante internet, es decir, la conexión de internet con más objetos que personas.

Este concepto fue propuesto por Kevin Ashton en 1999 cuando realizaba investigaciones en el campo de la identificación por radiofrecuencia (RFID) y sensores.

La idea era básicamente que una persona no podía tener el control o procesar toda la información que tiene a su alrededor en todo momento, así que, si las máquinas eran capaces de trabajar de manera autónoma procesando la información de los sensores se podría tener, por ejemplo, un control del stock de un almacén, un control de las medicinas caducas que se consumen en el mundo, un ahorro en las empresas, ciudades inteligentes capaces de priorizar el camino a los servicios de emergencias, controlar la iluminación o simplemente ir a recoger la basura solamente en aquellos sitios que estuviera llena, etc. Necesitando a las personas solamente para monitorizar su funcionamiento.

Por eso la IoT es un tema emergente que tiene importancia en los ámbitos técnicos, social y económicos. Actualmente tenemos móviles, sensores, automóviles, componentes de servicios públicos o industriales conectados a internet, y se prevé que en 2025 habrá hasta 100.000.000.000 de dispositivos conectados a la IoT con lo que cambiará la manera de trabajar y de vivir. Aun así, hay quienes en vez de oportunidades de trabajo o beneficios empresariales ven preocupaciones relacionadas con el tema de la seguridad o la privacidad. Esto implica retos jurídicos y políticos sobre cómo va afrontar la sociedad el hecho de estar “hiperconectado” (ver [1]).

Viendo esto se analizarán los aspectos más relevantes.

2.1.1. Seguridad y privacidad

La implementación de la IoT tiene el gran reto de hacer que la gente confíe en que los dispositivos conectados estén libres de vulnerabilidades, especialmente si tenemos sistemas de seguridad en casa, información médica en los hospitales o la información de los habitantes de una gran ciudad. Los dispositivos poco seguros pueden ser puntos de entrada para ataques y exponer la privacidad de los datos de los usuarios a nivel global. Por lo tanto, los desarrolladores y usuarios de dispositivos tienen la responsabilidad de asegurar que no se está exponiendo a los usuarios ni a la propia red.

2.1.2. Legalidad

La ley siempre va un paso por detrás de las innovaciones tecnológicas, por lo que en este caso habría que adaptar la ley para evitar el tráfico de datos entre empresas como por ejemplo, recoger un flujo de datos personales en una jurisdicción y para su procesamiento lo transmiten a otra jurisdicción donde las leyes de protección de datos son diferentes. Otro aspecto a tener en cuenta es el hecho de usar la información de los usuarios para realizar estudios que posteriormente tengan resultados discriminatorios para algunos.

2.1.3. Interoperabilidad

Un entorno lleno de implementaciones técnicas diferentes podría quitar valor para los usuarios y el sector industrial, por eso hay que tener estándares apropiados para que no haya un mal funcionamiento de los recursos de la red, genéricos, abiertos y disponibles para que haya más oportunidades de innovación.

Así bien, aunque no necesariamente tengan que tener las siguientes características, sí que el concepto de IoT es lo que pretende fomentar. A continuación enumeramos algunas de estas características para que un modelo pueda ser vinculado o asociado a la IoT:

- Poco mantenimiento y poca intervención humana (básicamente realizar monitorizado) una vez hechas las configuraciones pertinentes.
- Consumo bajo de energía (los sensores más utilizados utilizan tecnologías como bluetooth low energy o zigbee).
- Sensores de coste bajo y largo alcance.
- Interoperabilidad entre diferentes tecnologías de comunicación.
- Simplicidad de uso y configuración.
- Interacción de servicios entre objetos.
- Captura de información automática.

2.2. Ámbito para las aplicaciones IoT

El incremento exponencial de elementos conectados a internet hace que se tenga una gran variedad de ámbitos donde se pueda aplicar el concepto de IoT. A continuación se explicarán los ámbitos más importantes.

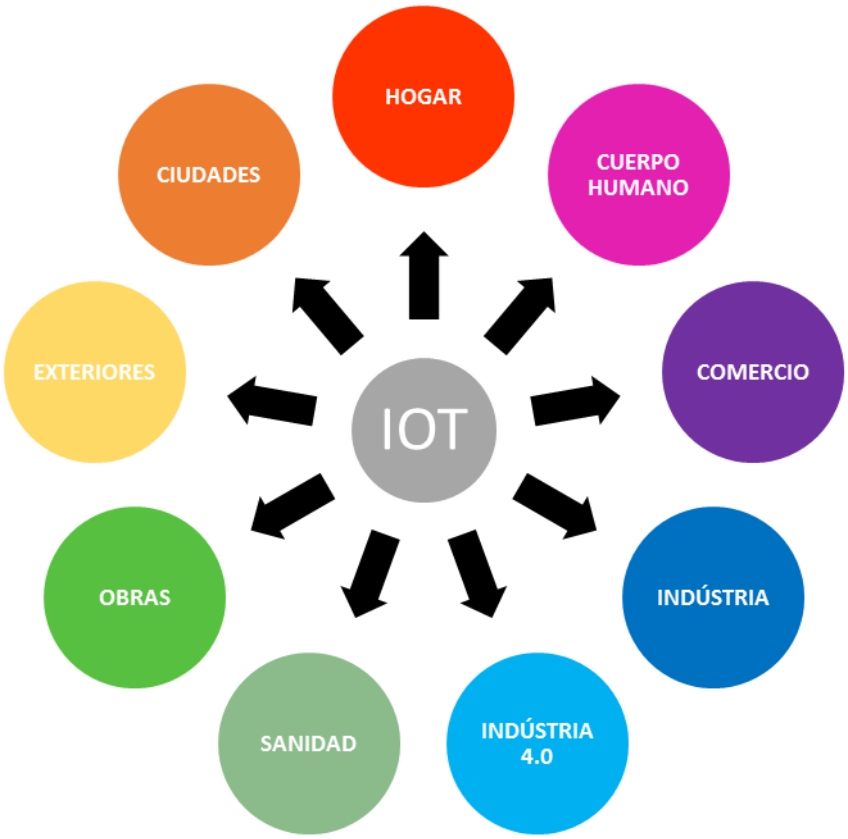

Fig. 2.1 Ámbitos en IoT

- **HOGAR:** Controladores y sistemas de seguridad para el hogar.
- **CUERPO HUMANO:** Son dispositivos que van unidos al cuerpo humano o dentro del mismo. Pueden ser desde prendas de ropa que contienen sensores para monitorizar el ritmo cardíaco, hasta sensores que detectan si necesitamos medicación.
- **COMERCIO:** Básicamente en todos los espacios comerciales hay sistemas de autopago por NFC u optimización del inventario mediante RFID.
- **INDÚSTRIA:** Granjas o fábricas son sitios donde hay rutinas de trabajo repetitivas y se busca la optimización del uso de los equipos y el inventario.
- **INDÚSTRIA 4.0:** Corresponde a la industria inteligente capaz de adaptarse a las necesidades, procesos de producción y asignar los recursos de manera eficiente.
- **SANIDAD:** Permite a los médicos hacer un seguimiento de sus pacientes monitorizando su salud en tiempo real o hacer un seguimiento y gestión de suministros y medicamentos.

- **OBRAS:** En la minería, petróleo, gas y construcción se busca la eficiencia operativa, mantenimiento predictivo, salud y seguridad.

- **EXTERIORES:** Los usos exteriores incluyen las vías de ferrocarril, los vehículos autónomos (fuera de los centros urbanos) y la navegación aérea, el enrutamiento en tiempo real, la navegación conectada, el seguimiento de envíos

- **CIUDADES:** Espacios públicos e infraestructura en entornos urbanos, sistemas de control adaptativo de tráfico, contadores inteligentes, monitoreo ambiental, gestión de recursos.

CAPÍTULO 3. PROTOCOLOS DE COMUNICACIÓN DE LAS IoT

Para definir protocolos de comunicación primero se tiene que entender las dos palabras que forman el concepto. Un protocolo son reglas y/o instrucciones necesarias para crear las bases básicas para la ejecución de una acción entre uno o varios individuos. La comunicación se entiende como el intercambio de información de manera activa y/o pasiva entre dos o más participantes.

Una vez se entienden estos dos conceptos, se puede entender el concepto de protocolos de comunicación, el cual se entiende como un conjunto de reglas que permite que se comuniquen dos o más entidades de un sistema (se intercambian información). En definitiva, se basa en las normas que tienen que seguir los componentes para comunicarse.

En una red en la que se comparten datos, los protocolos de comunicación son los encargados de controlar y dirigir las comunicaciones internas. De esta manera se pueden efectuar las comunicaciones entre dispositivos tecnológicos, sin tener problemas de entendimiento. Pero se ha de tener presente que todos los dispositivos tienen que poder entender el mismo protocolo de comunicación, aunque es posible que un dispositivo entienda más de un protocolo de comunicación.

Actualmente hay muchos protocolos que facilitan la posibilidad de crear un sistema de IoT, aunque no todos son igual de eficientes y resolutivos.

Estos protocolos se basan en el modelo TCP/IP. Para entender porque se basan en este modelo y no con el de la capa OSI, se explicara brevemente en que consiste este modelo.

El modelo TCP/IP se utiliza en comunicaciones de redes y provee la conexión de un servicio, extremo a extremo, especificando como tienen que ser los datos (direccionados, formateados, transmitidos y recibidos por el receptor).

El modelo TCP/IP se basa igual que en el modelo de OSI, en unas capas jerarquizadas. El objetivo principal de cada capa es proveer de servicios a las capas superiores sin ningún filtro entre ellas. De esta manera, las capas solo se preocupan del nivel inmediatamente inferior para solicitar servicios y del nivel inmediatamente superior para ofrecer servicios (dar resultados).

- Capa de aplicación: Se trata de los servicios de Internet que puede utilizar el usuario. Utilizan la capa inferior, la de transporte, para enviar y recibir datos.

- Capa de transporte: Se encarga de que los paquetes transmitidos lleguen sin errores y retransmite los paquetes perdidos.

- Capa de internet: Es conocida como capa de red, porque es la capa equivalente en el modelo OSI. Transfiere y acepta los paquetes para la red, que luego son utilizados por la capa superior (capa de transporte).

- Capa de acceso al medio: Se divide en dos partes, la más básica que identifica el tipo de protocolo a utilizar, en este caso TCP/IP, y el control de errores. La segunda parte, se trata del hardware que se utiliza en la red.

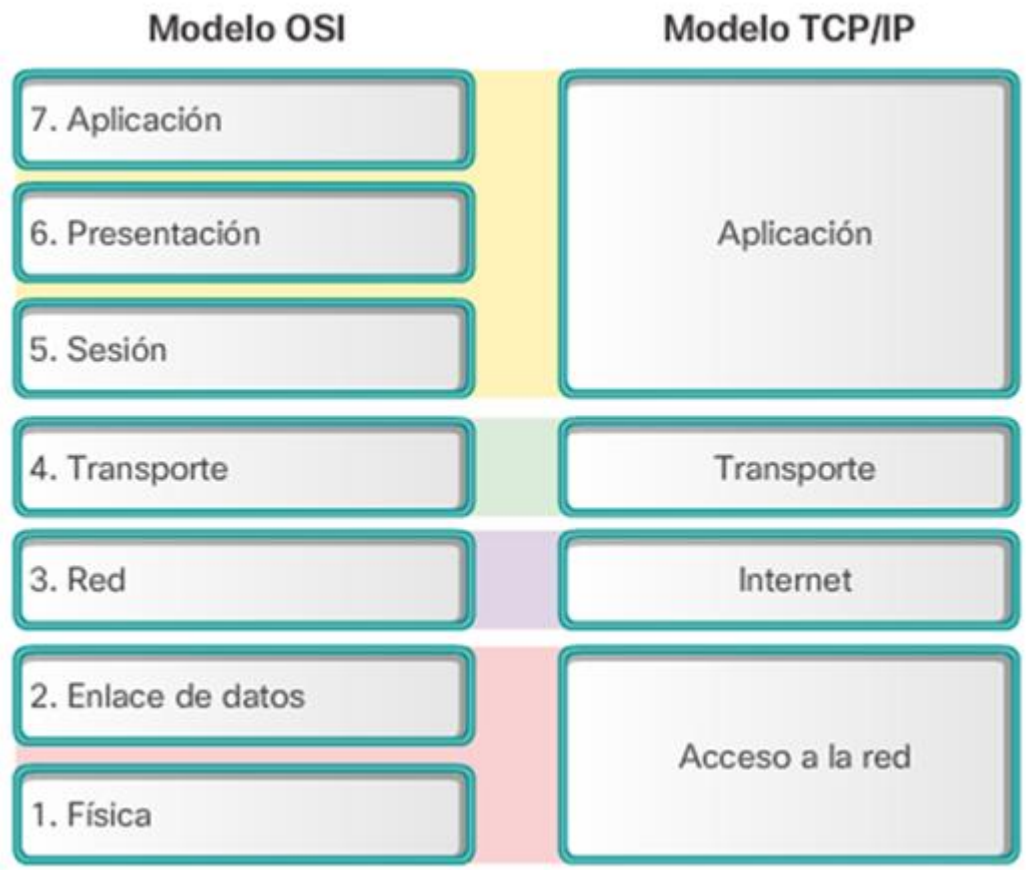

Fig. 3.1 Comparativa modelo OSI y TCP/IP

Los protocolos que se verán y se compararan con el MQTT se dividen en dos grupos: Protocolos utilizados en las IoT y Protocolos dedicados a las IoT (ver [8]).

3.1. Protocolos utilizados en las IoT

El primer grupo Protocolos utilizados en la IoT se encuentran tres grandes protocolos (HTTP, WebSocket y XMPP).

3.1.1. HTTP

Se define como el protocolo que sigue el esquema petición-respuesta entre un servidor y un cliente. En el ámbito de las IoT este protocolo se implementa con un solo cliente en el dispositivos, de esta manera solo se pueden realizar peticiones a un servidor Web y no recibir peticiones de conexión ya que no tenemos el servidor implementado. De esta manera se aumenta la seguridad de la red, las máquinas externas no pueden conectar con los dispositivos de la instalación IoT.

En resumen, HTTP es ideal para situaciones donde es necesario transmitir grandes cantidades de datos, pero si se trata de instalaciones con IoT no es rápido ni ligero.

3.1.2. WebSocket

Se basa en implementar una conexión (canal) bidireccional y full-duplex a través de un socket TCP. Se usa WebSocket junto con HTTP cuando los dispositivos soportan las cargas de HTTP, de esta manera se simplifica la complejidad de tener una conexión bidireccional y las dos partes (servidor-cliente) pueden empezar a enviar datos cuando se desee.

3.1.3. XMPP

Al ser un protocolo abierto y pensado para ser usado en mensajería instantánea lo hace un protocolo fácil de adaptar a las IoT. Principalmente para la administración de la comunicación entre dispositivos (microondas, nevera, calefacción, persianas, etc).

También es un protocolo seguro, escalable y direccional, que lo hace ideal para los consumidores de las IoT.

El problema de usar XMPP es que la mayoría de cortafuegos están configurados para dejar pasar solo el tráfico TCP dirigido al puerto que usa el protocolo HTTP, por lo tanto, XMPP tiene que utilizar HTTP para poder permitir el acceso a los usuarios que utilizan o están detrás de un cortafuegos (ver [9]).

3.2. Protocolos dedicados a las IoT

Este segundo grupo se basa en dos protocolos (CoAP, MQTT), son los protocolos más enfocados a los servicios ofrecidos por el IoT.

3.2.1. CoAP

El protocolo CoAP es el primero que veremos centrado en redes/aplicaciones IoT ya que está diseñado para funcionar con nodos, redes de baja potencia y perdidas.

El protocolo CoAP tiene fácil integración con la web debido a que está pensado para tener una fácil traducción al lenguaje HTTP. Al ser un protocolo fácil y simple de integrar lo hacen ideal para los aparatos que implementan el IoT.

Por tanto, si la instalación IoT utiliza un entorno Web, el protocolo CoAP es una gran elección ya que cubre todas las necesidades: conexión permanente, interfaz de programación fácil y protocolo fácil y ligero.

3.2.2. MQTT

El protocolo MQTT igual que CoAP es un protocolo de código abierto y por tanto todo el mundo lo puede usar y mejora, esto hace que sea más fácil de implementar en dispositivos usados para el IoT.

El MQTT es óptimo para redes de bajo ancho de banda, poco fiables, alta latencia y dispositivos restringidos. Por tanto, se basa en minimizar los requerimientos de recursos de los dispositivos conectados a la red, pero siempre manteniendo la máxima seguridad entre ellos, sin dejar que un dispositivo externo se puede conectar con facilidad (ver [7]).

De esta manera y como se está viendo en el estudio se ha decidido utilizar el protocolo MQTT para realizar la instalación de dispositivos IoT. Porque como se ha visto el MQTT se orienta a una instalación con muchos dispositivos básicos (clientes) que necesitan la supervisión de un dispositivos “master”. Aunque como cualquier protocolo de encaminamiento también tiene puntos negativos y en este caso es que los sistemas que usan este protocolo son lentos, se miden en segundo, por tanto no entran en el rango de “tiempo real”.

CAPÍTULO 4. PLACAS DE ELECTRÓNICA DE HARDWARE LIBRE

En este capítulo se explicara el significado, los diferentes tipos de placas con hardware libre que se pueden encontrar actualmente y el motivo por el que se decide escoger Arduino por delante de otras opciones.

Las placas de electrónica de hardware libre se basan en el concepto DIY. Este concepto se basa en hacer las cosas tú mismo y no depender de las piezas o dispositivos creados por las grandes empresas tecnológicas. Esta idea consigue reunir a los desarrolladores y emprendedores que prefieren usar materiales con código abierto para realizar sus propios diseños. De esta manera, se consigue crear una comunidad que comparte las tecnologías creadas por uno mismo.

La más utilizada es Arduino pero existen muchas más que se usan por la comunidad de desarrolladores de software libre, las cuales se verán a grandes pinzeladas a continuación (ver [11]).

4.1. Raspberry Pi

Es una tarjeta diseñada y desarrollada en el Reino Unido por la Fundación Raspberry Pi. El principal objetivo se trata de estimular la enseñanza de la informática en las escuelas.

No es exactamente un hardware libre, ya que se corresponde a un producto de marca registrada, pero permite el uso a nivel particular y educativo. En cambio, el software sí que es de código abierto.

Al ser un ordenador pequeño, aproximadamente igual de grande que una tarjeta de crédito, puede utilizar lenguajes de programación de alto nivel (C++, Python y Java). Por tanto, se puede utilizar para desarrollar inventos más complejos que Arduino.

4.2. Beaglebone

La placa BeagleBone funciona con Linux. Por este motivo, cualquier persona que desee desarrollar su propio software con esta placa puede utilizar la mayoría de lenguajes (Python, PHP, C++, Java, etc). Por otra parte, también es compatible con otros sistemas operativos como Ubuntu o Android.

Pero tiene el problema que se corresponde a una de las placas de hardware libre más caras.

4.3. Nanode

Desarrollado por el ingeniero electrónico del Reino Unido, Ken Broak, del grupo Hackerspace de Londres, una comunidad donde se comparten ideas y herramientas de desarrollo tecnológico.

Nanode se corresponde a la evolución directa de Arduino. Ya que permite conectarse a internet mediante una API y permite ser utilizado como servidor web. Se puede utilizar desde cualquier sistema operativo (Linux, Windows e incluso MAC).

4.4. Waspmote

Creado por la startup española Libelium. Se basa principalmente en solucionar los problemas de las IoT de como conectar dispositivos y recabar la información que generan.

Las placas que se comentan en este capítulo están basadas para ser usadas en escuelas, universidades o para todo tipo de desarrolladores. En cambio, esta placa esta destinada a ser usada en escenarios reales.

El principal problema respecto con Arduino es que no se trata de una placa muy conocida a nivel mundial. Pese a que utilizan el mismo entorno de desarrollo.

4.5. Arduino

Como las anteriores vistas en el capítulo, se basa en el desarrollo de una placa de hardware libre y es la más usada a nivel mundial por los desarrolladores de código libre.

Las características se explican con más detalle en el capítulo 5. Pero a grandes rasgos se compone de una placa de electrónica que incorpora varios pines hembra que permite las conexiones entre los sensores y el microcontrolador re-programable.

Como apunte final se tiene que tener en cuenta que se han diseñado varios modelos diferentes de placas. Por tanto, antes de obtener una se tiene que hacer un estudio previo para ver cual se ajusta más a las necesidades de cada proyecto. En el caso de este proyecto se ha decidido escoger la placa Arduino UNO.

CAPÍTULO 5. TECNOLOGÍAS IMPLEMENTADAS

5.1. MQTT

Antes de visualizar los motivos de porque se escoge el protocolo MQTT, se verá en más detalle las especificaciones de este protocolo.

MQTT significa “Message Queue Telemetry Transport” y como su nombre indica se basa en utilizar la mensajería publicación/suscripción entre máquinas en el concepto de “Internet of Things”.

Como el protocolo se enfoca, sobre todo, en la comunicación entre sensores, está pensado para ser simple, de fácil implementación, ligero y lo más importante, utilizar muy poco ancho de banda. De esta manera el coste computacional entre la comunicación entre máquinas es bajo y por lo tanto se necesitan pocos recursos (CPU, RAM, etc). Como punto negativo, al utilizar poco ancho de banda y pocos recursos tiene una alta latencia, medida en segundo. Aunque esto no es un problema cuando se trata de sensores ya que para el ojo humano es poco tiempo.

Por tanto, los objetivos de la creación del protocolo MQTT es minimizar el ancho de banda, el coste computacional, garantizar la fiabilidad, utilizar comunicación bidireccional entre máquinas y asegurar un grado alto de seguridad.

5.1.1. Seguridad del protocolo

Siempre se ha de partir de la base que cualquier máquina que esté conectada a internet nunca podrá garantizar una seguridad del 100%, por mucho que se intente, siempre habrá alguna vulnerabilidad. Partiendo de esta base el protocolo MQTT es bastante seguro, y soporta el cifrado mediante SSL, ya que, se ha de tener presente que se trata de un estándar ISO y tiene que cumplir ciertos requisitos.

Por tanto, se pueden enviar datos cifrados utilizando el protocolo MQTT y utilizando el cifrado SSL. Pero si se decide utilizar el cifrado el protocolo deja de ser simple y aumenta el coste computacional.

Cuando se habla de seguridad se ha de tener muy presente que el servidor puede estar en “cloud” (MQTT Server Cloud) o en físico (Mosquitto). La mejor opción cuando se trata de seguridad es utilizar el MQTT Server Cloud debido a que se actualiza de manera automática y se solucionan los problemas de seguridad que puedan tener las versiones anteriores. En cambio, si se utiliza el Mosquitto que tiene el servidor físico al no actualizarse de manera automática está la posibilidad de que se quede con una versión antigua.

Se puede pensar que utilizar el Mosquitto es una buena opción y que el usuario se encargará de realizar todas las actualizaciones a tiempo, pero esto no es cierto porque los usuarios quieren que todo sea “plug-and-play”, instalar el aparato pertinente y olvidarse por completo.

En conclusión, el protocolo MQTT es seguro y si se desea se puede aumentar la seguridad implementando el cifrado SSL, pero teniendo muy presente que se pierde la gran ventaja del protocolo, el bajo coste computacional.

5.1.2. Arquitectura

MQTT sigue una topología en forma de estrella donde el elemento (nodo) principal tiene la capacidad de gestionar un gran número de clientes. El nodo principal se lo conoce como “Broker” y los elementos conectados a este elemento se les conoce como clientes debido a que dependen en su totalidad al nodo principal (ver [3]).

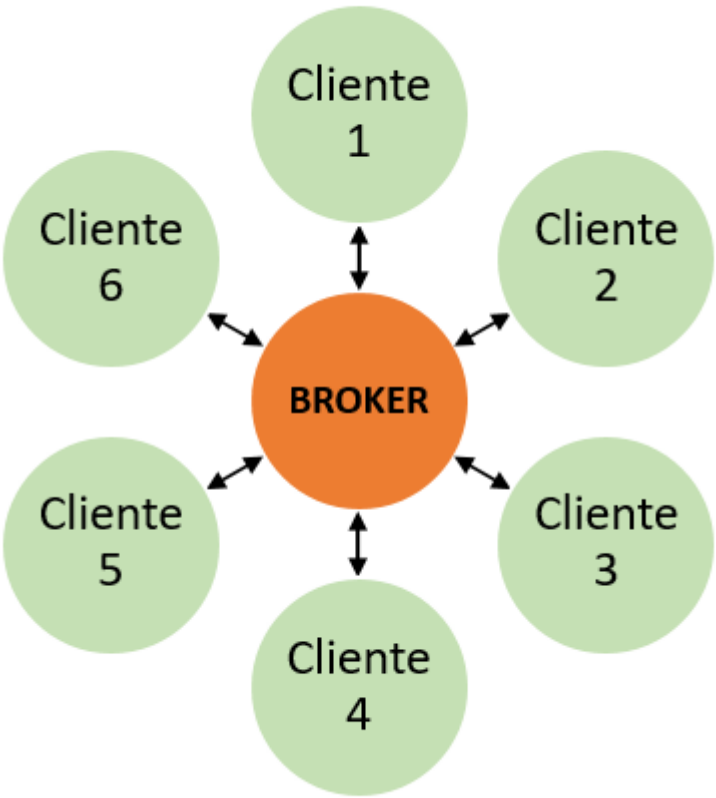

Fig. 5.1 Arquitectura MQTT

Entendiendo el nombre de los elementos se entiende que el “bróker” es el elemento encargado de transmitir los mensajes y gestionar la red, mientras que los clientes solo reciben y envían mensajes.

La principal característica de los clientes es que son independientes entre ellos. Por tanto, no necesitan saber si se encuentra alguien en el otro lado, simplemente transmiten la información. Esta característica permite crear proyectos con una mayor escalabilidad.

Hace falta puntualizar, antes de que se continúe con la explicación, que una característica a tener en cuenta es que el protocolo MQTT permite establecer las comunicaciones cifradas lo que nos aporta un extra de seguridad por si se diera la necesidad.

La arquitectura del MQTT se basa en el concepto publicación/suscripción. El “Broker” se encarga de distribuir los mensajes entre los receptores que se hayan suscrito al “topic” (tema en español), de esta manera los receptores (clientes) solo reciben los mensajes que van dirigidos a los “topics” que están suscritos.

Una de las ventajas de los “topics” es que se puede tener una estructura jerárquica, por tanto, se puede establecer relaciones padre-hijo, de esta manera si se está suscrito a un “topic” padre se puede recibir la información de sus hijos. Gracias a esto se pude pasar de una estructura en forma de estrella a una estructura jerárquica, como se puede observar en la siguiente figura:

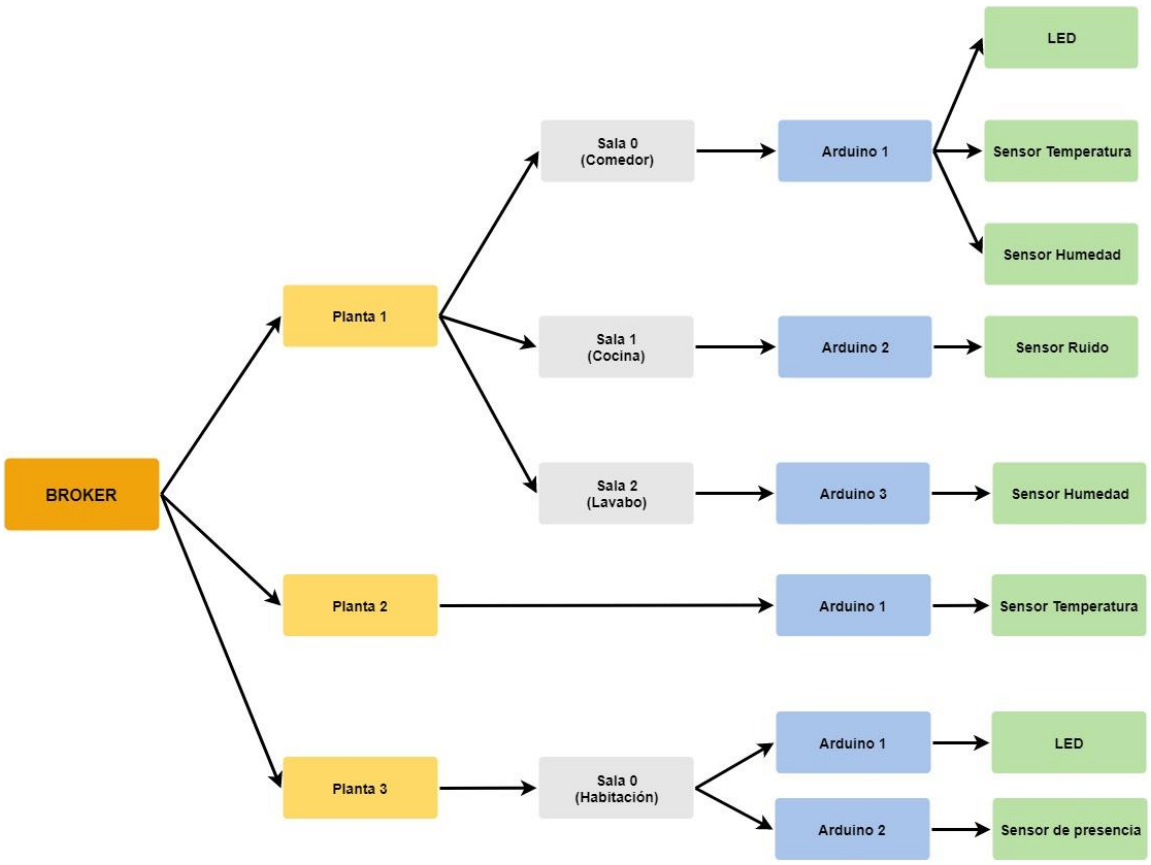

Fig. 5.2 Disposición de los sensores

El suscriptor que envía el mensaje en ningún momento sabe a quién va dirigido, solo lo sabe el “Broker”, que se dedica a distribuir el mensaje a los suscriptores que están suscritos al “topic” pertinente.

Pero la caracterísitca más imporante del protocolo MQTT son los mensajes que se transmiten. Se envían de forma asíncrona, ya que no se tiene que esperar ningún tipo de respuesta una vez enviado.

Los mensajes estan formados por tres partes imprecindibles:

- Encabezado fijo: es obligatorio que todos los mensajes contengan esta información, que solo ocupa 2 bytes.
- Encabezado variable: no es obligatorio que los mensajes contengan está información, ocupa 4 bits.
- Mensaje: En implementaciones reales ocupa entre 2 y 4 KB, pero teoricamente puede tener un máximo de 256 Mb

Pero como el protocolo esta diseñado para ocupar el menor ancho de banda, cada bit que se utiliza está estudiado cuidadosamente. Por tanto, y siguiendo las tres partes mencionadas anteriormente, el tamaño máximo del mensaje se decide por el tipo de implementación que se haya realizado y tiene un formato libre, lo decide el programador (ver [6]).

5.1.3. QoS

En este punto se explicaran los tres niveles de calidad del servicio que proporciona MQTT. Pero antes, para poder entender bien los tres niveles se definirán de manera breve el significado de QoS.

La calidad del servicio (QoS) sirve para para calificar el rendimiento de una red visto des del punto de vista de los usuarios. Es importante ya que mide los servicios de la red, como, tasa de error, rendimiento, ancho de banda, retraso de la transmisión, etc. En resumen, se refiere a la calidad del servicio en la conexión entre un cliente y el broker.

Antes de empezar a explicar los tres niveles, se debe tener en cuenta que el nivel de QoS está restringido a la comunicación de cliente a servidor o de servidor a cliente. Por tanto, si se publica un mensaje con nivel 1 de QoS, la calidad de este servicio solo se aplica entre este cliente y servidor. En ningún caso entre otros clientes y servidores. Por otro lado, el que determina el nivel de QoS siempre se trata del cliente (ver [10]).

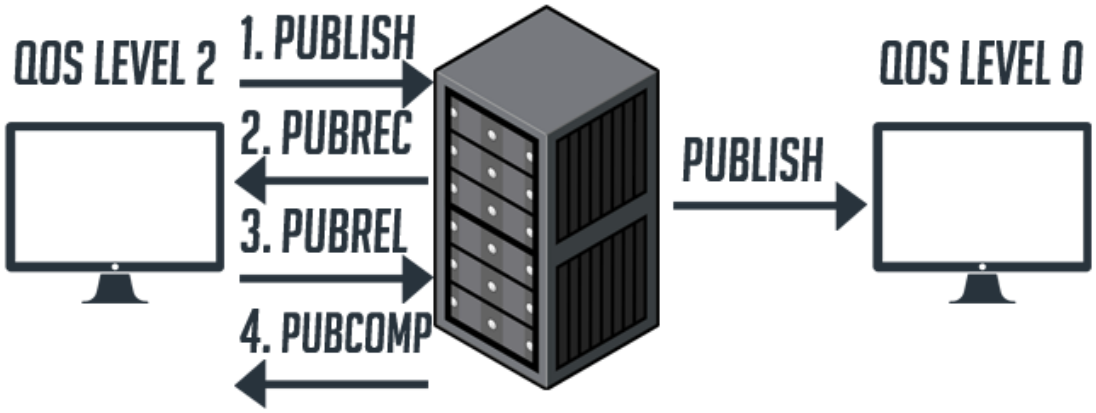

Fig. 5.3 QoS diferentes niveles (ver [10])

5.1.3.1. MQTT QoS Nivel 0 – Una vez máximo

Se trata del nivel más simple y fiable de los tres que existen. El cliente envía el mensaje solo una vez y el servidor no tiene que contestar diciendo que lo ha recibido, de esta manera se evita que el servidor tenga que almacenar el mensaje recibido. Por tanto, no existe ninguna garantía que el servidor reciba el mensaje. Por otra parte, el cliente tampoco almacena el mensaje una vez enviado, de esta manera se evita tener mensajes duplicados.

Los momentos para usar este nivel, por ejemplo, son:

- Los mensajes se tienen que entregar de manera rápida.
- El internet es fiable.
- Si se pierden mensajes no importa.

5.1.3.2. MQTT QoS Nivel 1 – Al menos una vez

En este nivel se pueden duplicar mensajes. Debido a que cuando el cliente envía un mensaje al servidor, este tiene que enviar un mensaje de confirmación (PUBACK) al cliente. De esta manera el cliente sabe que el servidor ha recibido el mensaje, ya que si no recibe el mensaje PUBACK reenviará otra vez el mensaje original.

Una vez el servidor recibe el mensaje tiene que enviarlo a todos los clientes que estén suscritos al topic (tema) pertinente y espera el mensaje PUBACK del cliente. De igual manera que antes, si el servidor no recibe el mensaje PUBACK vuelve a enviar el mensaje.

En todos los casos, si el remitente no recibe el mensaje de confirmación PUBACK dentro de un periodo de tiempo, vuelve a publicar el mensaje original a los destinatarios pertinentes, pero esta vez con un indicador de duplicación

(DUP). El indicador DUP no se procesa por el cliente o el servidor, pero se pude usar, si se desea, por el programador para gestionar los mensajes duplicados.

En este nivel se garantiza que los mensajes son recibidos, pero si el PUBACK no se recibe a tiempo los mensajes se vuelven a enviar y por tanto se duplican. Esto hace que este nivel de QoS sea más lento que el nivel 0.

Los momentos para usar este nivel, por ejemplo, son:

- El servidor y/o cliente tiene que recibir todos los mensajes.
- Si existen mensajes duplicados se pueden gestionar de manera correcta.

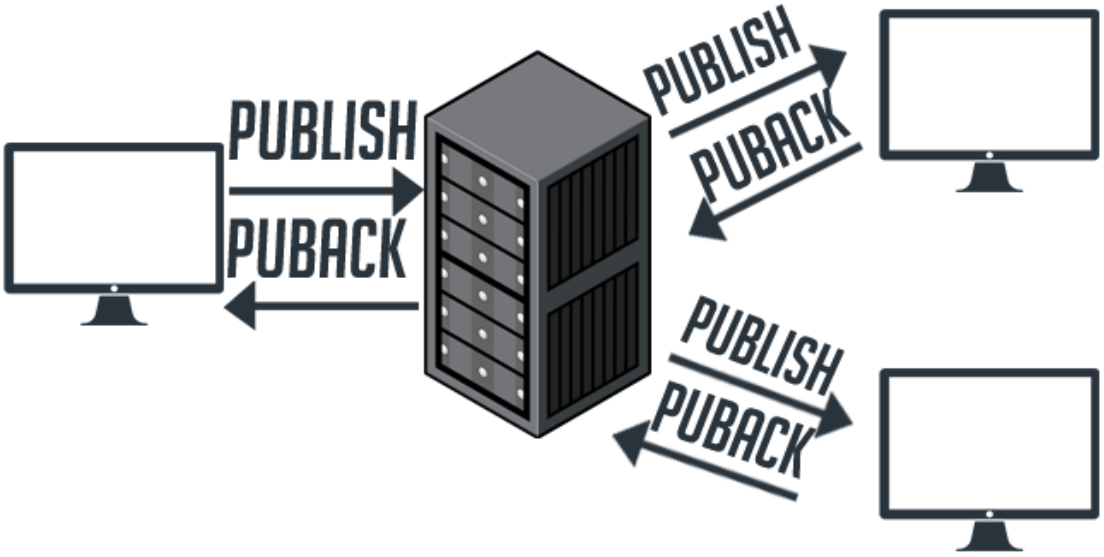

Fig. 5.4 QoS nivel 1 (ver [10])

5.1.3.3. MQTT QoS Nivel 2 – Exactamente una vez

A diferencia del nivel 1, este nivel de QoS garantiza que solo se reciba un mensaje. De esta manera no se tiene que gestionar los mensajes duplicados, ya que no existirán.

Este nivel se separa en cuatro pasos:

- Primer paso: Una vez el receptor recibe el mensaje, responde con un mensaje Publish Received (PUBREC) que confirma que se ha recibido el mensaje correctamente. Si el receptor no envía el mensaje PUBREC, entonces el emisor vuelve a publicar el mensaje con el indicador DUP.
- Segundo paso: El remitente elimina el primer mensaje de publicación justo después de recibir el mensaje PUBREC. Entonces envía un mensaje Publish Release (PUBREL) al receptor.

- Tercer paso: Una vez el receptor recibe el PUBREL, envía un mensaje Publish Complete (PUBCOMP) al remitente. Pero si el remitente no recibe el PUBCOMP, entonces vuelve a reenviar el mensaje PUBREL.
- Cuarto y último paso: El remitente elimina el mensaje de la cola.

Este nivel se trata del más lento de todos, por la cantidad de pasos que se tiene que seguir. Pero tiene las ventajas de que se revive el mensaje y no se duplica.

Los momentos para usar este nivel, por ejemplo, son:

- No hace falta entregar los mensajes de manera rápida.
- Los mensajes duplicados causan problemas y son difíciles de gestionar.

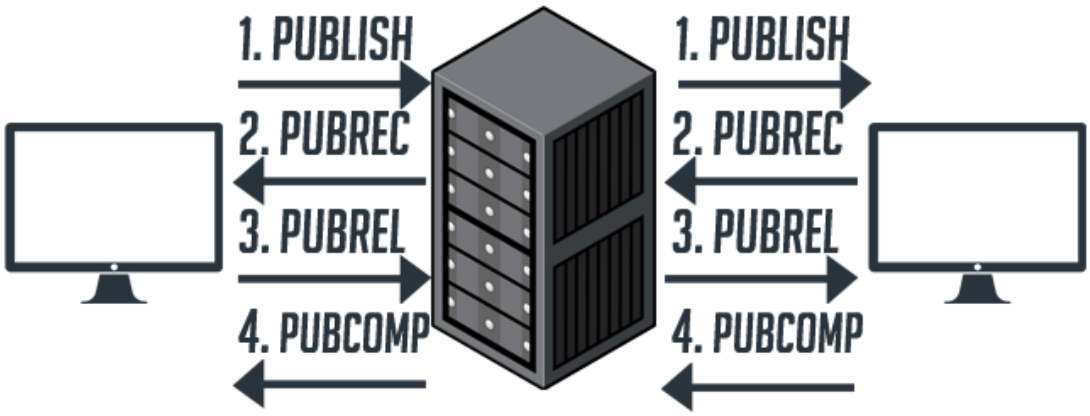

Fig. 5.5 QoS nivel 2 (ver [10])

5.1.4. Motivos de porque se escoge MQTT respecto los otros

MQTT es un protocolo de red que ofrece un gran equilibrio en las redes basadas en el “Internet of Things”.

Permite a los desarrolladores utilizar este protocolo en redes que tienen un ancho de banda limitado y que necesitan gran flexibilidad (sistema asíncrono).

Pero el motivo más importante para escoger el protocolo MQTT es el modelo que utiliza de publicación/suscriptor, el cual, se trata de separar al consumidor y al editor de los datos. De esta manera las interacciones que se producen entre publicadores y suscriptores las controlan los intermediarios. Un sistema basado en este modelo permite tener más de un suscriptor, necesario si se habla de sensores, y más de un publicador.

5.2. Arduino

El proyecto 'Arduino' comenzó en 2005 por el estudiante del instituto IVRAE Massimo Banzi, donde la idea del proyecto era básicamente dar un microcontrolador a los estudiantes de computación y electrónica a un precio mucho más asequible del que había, ya que se usaba el microcontrolador BASIC Stamp, cuyo precio era de 100$.

El primer prototipo de Arduino era una simple placa con circuitos eléctricos donde había conectados un microcontrolador muy simple y resistencias que servían para conectar otras resistencias o LEDS y algunos sensores. Es decir, todavía no contaba con el apoyo de ningún lenguaje de programación para el procesador de la Arduino. Posteriormente, se unieron al proyecto Arduino David Cuartielles, que ayudó a mejorar la interfaz de hardware dotándola de los microcontroladores y memoria necesarios para poder programarlos en código abierto, y Tom Igoe, que ayudó haciendo todavía más potente la placa y añadiendo puertos USB para poder conectarla al PC.

Finalmente Arduino tal y como lo conocemos estaba listo y empezaron a distribuirlo de forma gratuita por las facultades de electrónica y posteriormente la comercializaron.

Lejos estaba de imaginarse que años después este microcontrolador sería el líder de las tecnologías DIY (Do It Yourself). Esto significa que puedes comprar placas ensambladas, placas de expansión, kits y accesorios para personalizar tus proyectos de manera sencilla.

A continuación se muestra un esquema de la placa Arduino:

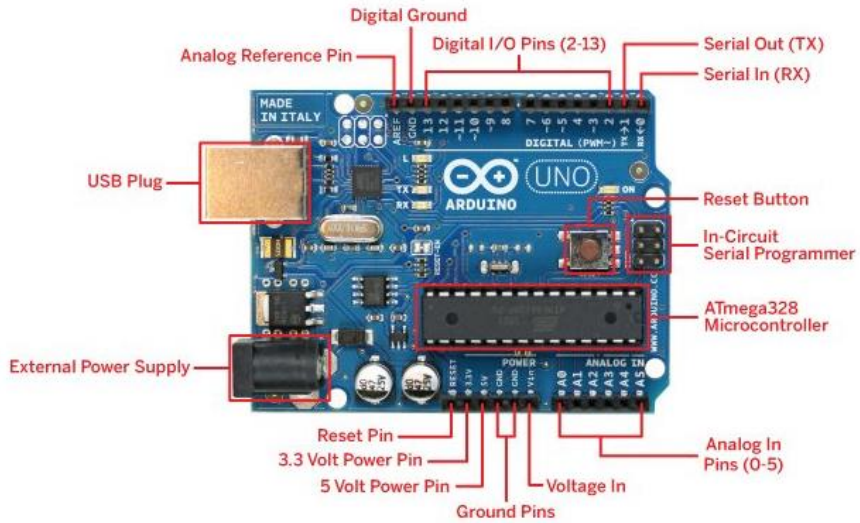

Fig. 5.6 Esquema Arduino UNO (ver [2])

Ahora existen varios tipos de microcontroladores 'Arduino' que se adaptan a las necesidades de cada proyecto de los usuarios y es por este motivo que nosotros hemos escogido la Arduino UNO. Es fácil de programar ya que el entorno de programación es el Arduino IDE y el lenguaje usado es C. Por otro lado, también existen varias páginas web o programas que te permiten hacer una programación por bloques, lo que todavía facilita más su uso.

Para un proyecto de IoT es necesario conectar uno o varios microcontroladores a distintos sensores distribuidos en varias localizaciones, ya sea para implementar la domótica de una vivienda, para tener un campo de cultivo inteligente, sistemas de seguridad, etc.

Todos estos proyectos tienen en común que requerirán de varios sensores distribuidos en varias localizaciones y en algunos casos se podrá hacer la instalación mediante cableado y en otros casos no habrá la opción de tener tanto cableado por lo que la comunicación de algunos sensores o entre varios microcontroladores será mediante WIFI o bluetooth. Aquí es donde filtramos los modelos candidatos para nuestro proyecto y nos encontramos que cogiendo como base este microcontrolador, podemos ir añadiendo módulos en función de lo que necesitemos, es decir, si el entorno es un espacio cerrado y grande donde el WIFI no llegue con suficiente potencia podremos añadir a nuestro microcontrolador Arduino un módulo ethernet. En cambio, si el entorno permite que se use WIFI, podemos añadir un módulo WIFI.

Arduino es una placa con un microcontrolador de la marca Atmel con toda la circuitería de soporte conectado a un módulo adaptador USB-Serie que permite programar el microcontrolador desde cualquier PC usando el Arduino IDE de manera fácil.

Tiene 14 pines que se pueden configurar como entrada o como salida a los que se les puede contar cualquier dispositivo que sea capaz de transmitir o recibir señales digitales entre 0 y 5V.

También dispone de entradas y salidas analógicas, donde por las entradas podemos obtener los datos de los sensores mediante las variaciones de tensión. Las salidas son para enviar señales de control.

Como se ha dicho anteriormente, los 14 pines digitales se pueden usar de entrada y salida, y cada pin funciona a 5V y puede suministrar hasta 40mA que es el máximo. Además todos los pines disponen de una resistencia de pull-up interna de 20KΩ a 50KΩ que está desconectada excepto cuando indiquemos lo contrario por código.

Además tiene 6 pines de entrada analógicos que envían las señales a un conversor A/D de 10 bits.

A continuación se dispondrá a realizar una breve explicación de los principales pines de entrada y salida:

- RX y TX: Sirven para las transmisiones en serie de señales TTL
- Pines 2 y 3: se usan para las interrupciones externas en el atmega. El principal objetivo es que cuando estos pines detectan que hay un valor bajo o flancos de subida y/o bajada de la entrada se disparan.
- PWM: hay 6 salidas destinadas a la generación de señales PWM de hasta 8 bits.
- Pines 10, 11, 12 y 13: se usan para llevar a cabo las comunicaciones SPI, son las que se encargan de trasladar la información full dúplex en un entorno basado en maestro/esclavo.
- I2C: se encarga de establecer comunicaciones a través de un bus I2C. El bus I2C se puede encontrar en muchos dispositivos que utilizan esta misma interfaz, como podrían ser sensores, pantallas LCD, etc. Como apunte final del bus I2C, destacar que es un producto de Phillips para interconexiones de sistemas embebidos.

La Arduino se puede alimentar de dos formas distintas, ya sea usando el propio cable USB o utilizando una fuente de alimentación externa, en este caso, se podría usar una pila de 9V o un pequeño transformador, los límites para no dañar la placa se encuentran entre los 6 y 12V. Pero se ha de tener en cuenta que si se decide utilizar una alimentación menor a 7V, por eso no se recomienda, la salida del regulador de tensión a 5V puede dar menos voltaje. Como contra si se supera los 12V existe la posibilidad de que se dañe la placa.

La conexión de la alimentación se puede realizar mediante un conector de 2,1mm con el dispositivo en el centro o hacerlo directamente en los pines Vin y GND que se encuentran marcados sobre la placa.

Para finalizar la explicación de la alimentación de la Arduino, se ha de tener en cuenta que se puede medir el voltaje que hay en el Jack directamente desde el pin Vin. Excepto si se está alimentado la placa Arduino mediante el cable USB, entonces no se podrá monitorizar el voltaje que se tiene mediante el pin Vin (ver [5]).

Tabla 5.1 Características técnicas Arduino UNO (ver [2])

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada- salida digital	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

5.2.1. Tipos de Sensores para microcontroladores Arduino

Un sensor es un dispositivo que detecta magnitudes físicas o químicas y las transforma en señales eléctricas.

Las magnitudes químicas que tienen que detectar pueden ser por ejemplo: intensidad lumínica, desplazamiento, temperatura, presión, aceleración, fuerza, distancia, humedad, pH, etc.

Las magnitudes eléctricas que tienen que detectar pueden ser por ejemplo: una capacidad eléctrica (sensor de humedad), una corriente eléctrica (fototransistor), etc.

I en función de los datos que tenemos a la salida los sensores se pueden clasificar en analógicos o digitales.

Los sensores digitales son los que nos dan una señal con dos estados ‘1’ o ‘0’. En caso de que el sensor use comunicación por bus, habrá que implementar hardware adicional que nos proporcione una interfaz entre el bus y el Arduino, o utilizar alguno de los buses que tiene Arduino.

Los sensores analógicos son los que nos dan una señal variable en voltaje proporcional a la corriente que circula por el sensor. Un ejemplo sería un sensor de efecto hall que mide corrientes eléctricas que circulan a través del chip (ver [4]).

5.2.2. Características de los sensores

Las respuestas de los sensores se dividen en estáticas y dinámicas. Las estáticas son aquellas en que los instrumentos usados no presentan variaciones bruscas en magnitud cuando estos están midiendo cantidades estables. En cambio, las respuestas dinámicas son aquellas en que cuando se está midiendo una cantidad puede sufrir una variación en un momento concreto y entonces, es necesario conocer el comportamiento dinámico del instrumento en cuestión.

5.2.2.1. Características estáticas

- **Rango de medida:** Conjunto de valores que puede tomar la señal de entrada comprendidos entre el máximo y el mínimo detectados por el sensor con una tolerancia de error aceptable.
- **Precisión:** Error de medida máximo esperado.
- **Offset:** Valor de la variable de salida cuando la variable de entrada es nula. Si el rango de medida no llega a valores nulos de la variable de entrada, se establece otro punto de referencia para definir el offset.
- **Linealidad o correlación lineal:** Expresa lo constante que resulta la sensibilidad del sensor.
- **Sensibilidad:** Variación de la salida producida por una variación de entrada. Pendiente de la curva de calibración. Cuanto mayor, mejor.
- **Resolución:** Menor cambio en la magnitud de entrada que se aprecia en la magnitud de salida.
- **Rapidez de respuesta:** Puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.
- **Derivas:** Son otras magnitudes, aparte de la medida como magnitud de entrada, que influyen en la variable de salida. Por ejemplo, pueden ser condiciones ambientales, como la humedad, la temperatura u otras como el envejecimiento (oxidación, desgaste, etc.) del sensor.
- **Repetitividad:** Error esperado al repetir varias veces la misma medida.

- **Histéresis:** Diferencia entre valores de salida correspondientes a la misma entrada, según la trayectoria seguida por el sensor.

- **Exactitud:** Diferencia entre el valor de la salida real y el teórico de esta salida.

5.2.2.2. Características dinámicas

- **Velocidad de respuesta:** Es la capacidad de que la señal de salida siga sin retraso las variaciones de la señal de entrada.

- **Respuesta frecuencial:** Relación entre la sensibilidad y la frecuencia cuando introducimos una señal senoidal a la entrada, cuya representación se puede hacer con un diagrama de Bode.

- **Estabilidad:** Es la desviación a la salida del sensor al variar parámetros externos de los que se pretende medir.

CAPÍTULO 6. IMPLEMENTACIÓN

6.1. Tipos de escenarios dependiendo del broker

Se puede diseñar un entorno en local y otro en la nube en función de las necesidades de cada usuario. Por ello a continuación se presentan dos esquemas, uno en local y otro en la nube, cada uno con unas características diferentes:

6.1.1. Broker MQTT Local

En un entorno local se pretende que la información de nuestros sensores sea procesada y enviada por un broker local que en este caso será una Raspberry con un servidor mosquitto a un dispositivo móvil o un PC al cuál tendremos control total y acceso. Eso significa que la información nunca sale de nuestro entorno por lo que se pretende que nadie de fuera tenga acceso a dicha información.

A continuación se muestra un primer diseño donde hay 3 Arduino con distintos sensores conectados a ellas, donde se comunican con un Broker MQTT que en este caso es una Raspberry Pi con Mosquitto, ya que es el servidor más usado.

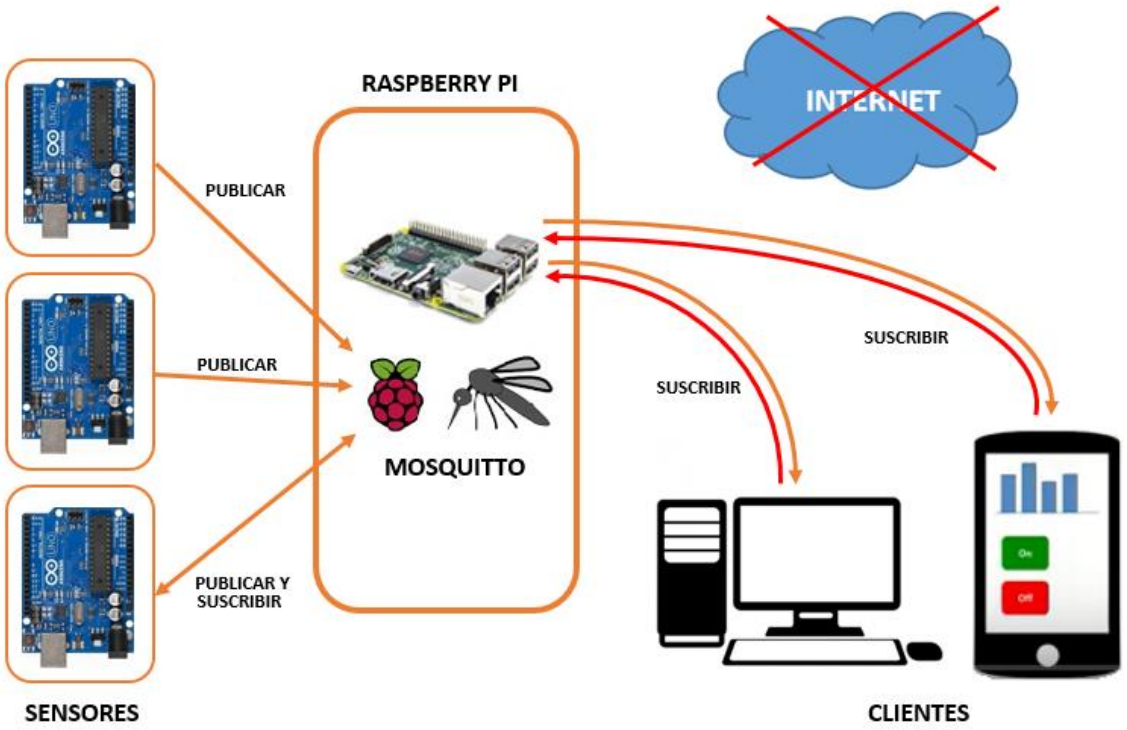

Fig. 6.1 Broker en local

En este escenario se puede simular por ejemplo, que la Raspberry está en el comedor y tiene 3 sensores que están conectados cada uno a una Arduino (presencia, iluminación, temperatura/humedad). Toda la información que recolectan los sensores es publicada al broker (Raspberry), y en función de quién se haya suscrito al topic de comedor/temperatura_humedad, comedor/presencia o comedor/iluminación recibirá la información de uno, dos o los tres sensores. Como el escenario es en un servidor local la información nunca saldrá al exterior protegiendo los datos de los usuarios. A continuación se muestran los pasos de la comunicación:

- Se supone que el servidor (broker) está funcionando y los sensores publican en un topic que se ha declarado en el broker. El primer sensor publica en el topic comedor/temperatura_humedad, el segundo en el topic comedor/presencia y el tercero en el topic comedor/iluminación. Esto significa que cada cierto tiempo (según se haya configurado) mandan la información que reciben los sensores hacia el broker.

- Tanto el PC como el móvil se suscriben a los topics de los cuales quieren recibir la información. Por ejemplo, el PC se suscribe a los topics comedor/temperatura_humedad y comedor/iluminación y el móvil se suscribe al topic de presencia. A partir de ahora, cada dispositivo solo recibirá la información a la cual está suscrita.

6.1.2. Broker MQTT en la nube

En un entorno en la nube se pretende que la información de nuestros sensores sea procesada y enviada por un broker que está en un servidor por lo que hará falta tener conexión a internet. Esta información será enviada a un dispositivo móvil o un PC igualmente pero en este caso como el servidor es web y de terceros no tendremos control sobre lo que está haciendo el broker.

A continuación se muestra un primer diseño donde hay 3 Arduino con distintos sensores conectados a ellas, donde se comunican con un Broker MQTT en la nube, por ejemplo el cloud MQTT, y este enviará la información al PC y al móvil.

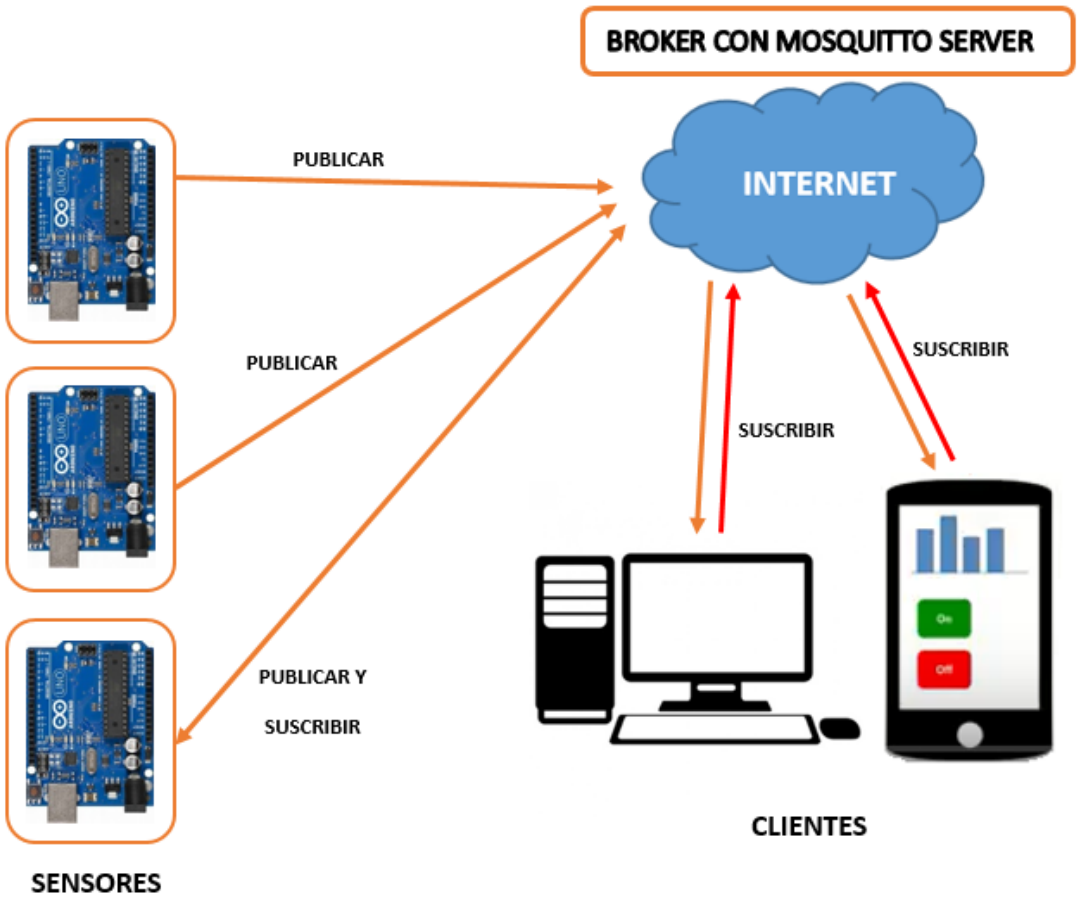

Fig. 6.2 Broker en la nube

En este escenario se puede simular de igual forma el caso anterior (en local) y toda la información que recolectan los sensores es publicada al broker (en la nube) mediante ethernet o WIFI, y en función de quién se haya suscrito al topic de comedor/temperatura_humedad, comedor/presencia o comedor/iluminación recibirá la información de uno, dos o los tres sensores. Como el escenario es en un servidor de la nube, la información es almacenada y procesada por un servidor de terceros . A continuación se muestran los pasos de la comunicación:

- Se supone que el servidor de la nube (broker) está funcionando y los sensores publican en un topic que se ha declarado en el broker. El primer sensor publica en el topic comedor/temperatura_humedad, el segundo en el topic comedor/presencia y el tercero en el topic comedor/iluminación. Esto significa que cada cierto tiempo (según se haya configurado) mandan la información que reciben los sensores hacia el broker.
- Tanto el PC como el móvil se suscriben a los topics de los cuales quieren recibir la información. Por ejemplo, el PC se suscribe a los topics comedor/temperatura_humedad y comedor/iluminación y el móvil se suscribe al topic de presencia. A partir de ahora, cada dispositivo solo recibirá la información a la cual está suscrita.

Para el proyecto se ha utilizado un esquema con un broker en la nube que recibe los datos de una arduino uno que tiene conectados un sensor de temperatura/humedad y un led. El esquema se basa en la realización del esquema de la figura 5.2 para ver cómo se realiza la comunicación entre los sensores, el broker y el cliente que en este caso serán dos: un PC y un móvil.

6.2. Componentes:

6.2.1. Arduino UNO / Ethernet Shield

Como se ha comentado en el apartado de teoría 4.2 se ha utilizado una Arduino UNO como microcontrolador y para realizar la conexión con el broker se hará mediante Ethernet por lo que se ha acoplado a la placa Arduino una placa ethernet que se conecta mediante un conector estándar RJ45 al router, y ésta es la parte física que implementa la pila de protocolos TCP/IP. También esta placa ethernet dispone de unos conectores que permiten conectar a su vez otras placas encima y apilarlas sobre la placa Arduino. Para comunicarse con la placa ethernet, Arduino UNO utiliza los pines digitales 10, 11, 12, y 13 (SPI), por lo que no se podrán usar posteriormente.

Conectadas quedan de la siguiente manera:

Fig. 6.3 Arduino UNO ensamblada a la Ethernet Shield

Igual que Arduino esta placa ethernet cuenta con varios leds de información:

- **ON:** indica que la Arduino y la placa ethernet están alimentadas.
- **LINK:** indica la presencia de un enlace de red y parpadea cuando la placa envía o recibe datos.
- **100M**: indica la presencia de una conexión de red de 100 Mb/s.
- **RX**: parpadea cuando la placa ethernet recibe datos.
- **TX:** parpadea cuando la placa ethernet envía datos.

Una vez conectadas la Arduino y la placa ethernet, para cargar el código se hace exactamente igual, ya que es a través de la Arduino mediante el puerto USB.

Sus características son las siguientes:

- Opera a 5V suministrados desde el pin de 5V de la placa de Arduino.
- El controlador ethernet cuenta con 16K de buffer interno y no consume memoria.
- La placa ethernet se comunica con el microcontrolador por el bus SPI, por lo tanto para usarlo siempre habrá que incluir la libreria <SPI.h>, y para poder configurarla se deberá importar la librería <Ethernet.h>.
- Soporta hasta 4 conexiones simultáneas.

Por los pines 8 y 2 de la Arduino irán contados el led y el sensor de temperatura/humedad (dht11) respectivamente. A continuación se describen sus características:

6.2.2. LED

Un led es un diodo emisor de luz, es decir, emite luz al ser atravesado por una corriente eléctrica. Al ser la unión entre dos semiconductores con dopados distintos hace que se genere una barrera de potencial que hace que la corriente sólo pueda ir en uno de los dos sentidos. El sentido de la corriente se conoce porque el led tiene una pata más corta que la otra como se puede ver en la siguiente imagen:

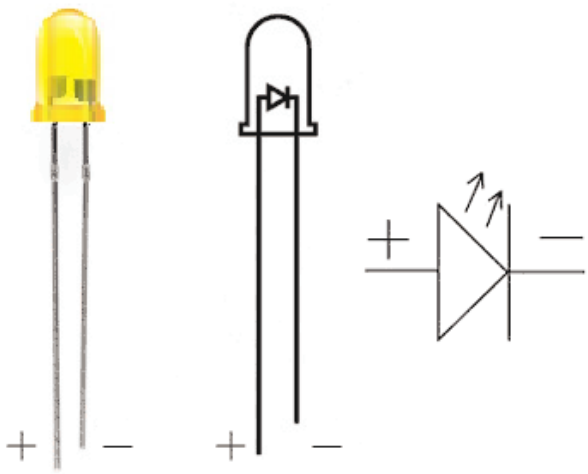

Fig. 6.4 Led

Otra consecuencia de la barrera de potencial es que incluso con la polaridad correcta, puede ser que no se esté dando la tensión correcta ya que para que los electrones atraviesen el dispositivo es necesario alcanzar un mínimo de tensión llamada tensión de polarización directa. A partir de este umbral los electrones atraviesan la barrera de potencial y como consecuencia el led se ilumina.

Se ha utilizado un led amarillo, es decir, con una longitud de onda de 565nm y 590nm y su tensión de polarización directa es aproximadamente 2.2 V.

6.2.3. DHT11 (Sensor de temperatura y humedad)

Una de las ventajas del DHT11 es que aparte de ser un sensor de temperatura y humedad a la vez, es que es digital a diferencia del LM35 que es más conocido.

Esto significa que se está más protegido frente al ruido a la hora de enviar la información.

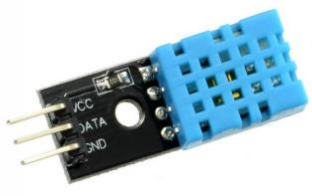

Fig. 6.5 Sensor de temperatura DHT11

Para este proyecto concretamente se ha utilizado un DHT11 insertado en una PCB, que tiene una resistencia de pull-up de 5kΩ y un led que avisa de que está en funcionamiento. En vez de 4 pines, este lleva 3 (Alimentación, Tierra, Datos) y sus características son las siguientes:

Tabla 6.1 Sensor DHT11

SENSOR DHT11	
Alimentación	de 3,5 V a 5 V
Consumo	2,5 mA
Señal de salida	Digital
Temperatura	
Precisión	a 25ºC ± 2ºC
Resolución	1ºC (8-bit)
Humedad	
Rango	de 20% RH a 90% RH
Precisión	entre 0ºC y 50ºC ± 5% RH
Resolución	1% RH

Aunque se conecte a un pin digital, se trata de un dispositivo analógico, por lo que la conversión A/D se hace en el sensor antes de ser enviada a la Arduino UNO. Así pues, la trama de datos que envía es de 40bits distribuidos de la siguiente manera:

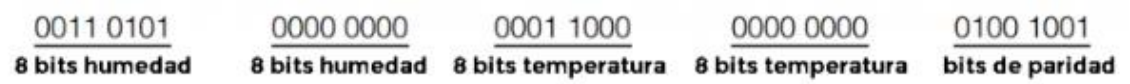

Fig. 6.6 Trama de datos que envía el sensor DHT11

Los primeros 8 bits son la parte entera del valor de humedad y los 8 bits siguientes son la parte decimal del valor de humedad. De igual manera, los siguientes 8 bits son la parte entera del valor de la temperatura y los 8 bits siguientes son la parte decimal del valor de la temperatura.

Finalmente los últimos 8 bits son los de paridad, es decir, es la suma de los otros 4 grupos de bits para asegurarse de que no hay datos corruptos desde que se envía la información a la Arduino UNO hasta que se recibe.

6.3. Esquema circuito

A continuación se muestra un esquema de la conexión de los componentes hecha con el programa Fritzing donde se aprecia que está la Arduino UNO debajo de la placa ethernet. Es aquí donde se realizan las conexiones de los componentes.

El led va conectado al pin 8 que es por donde se alimenta, además, se le conecta una resistencia en paralelo para no estropear el led, porque aunque la corriente de salida de los pines está limitada a 20mA es suficiente para acortar la vida del led. Finalmente se conecta a tierra por el pin de tierra.

El sensor de temperatura y humedad DHT11 va alimentado por el pin de 5V. En este caso como el sensor está insertado en una PCB y ya tiene una resistencia interna no hace falta conectarlo a una resistencia como el led. Otra salida del DHT11 va a tierra, y finalmente la transmisión de datos se realiza por el pin 2.

Para el envío de información al broker de la nube se realizaría una conexión a internet por el puerto ethernet, el cual tendría conectado un cable que vendría del router.

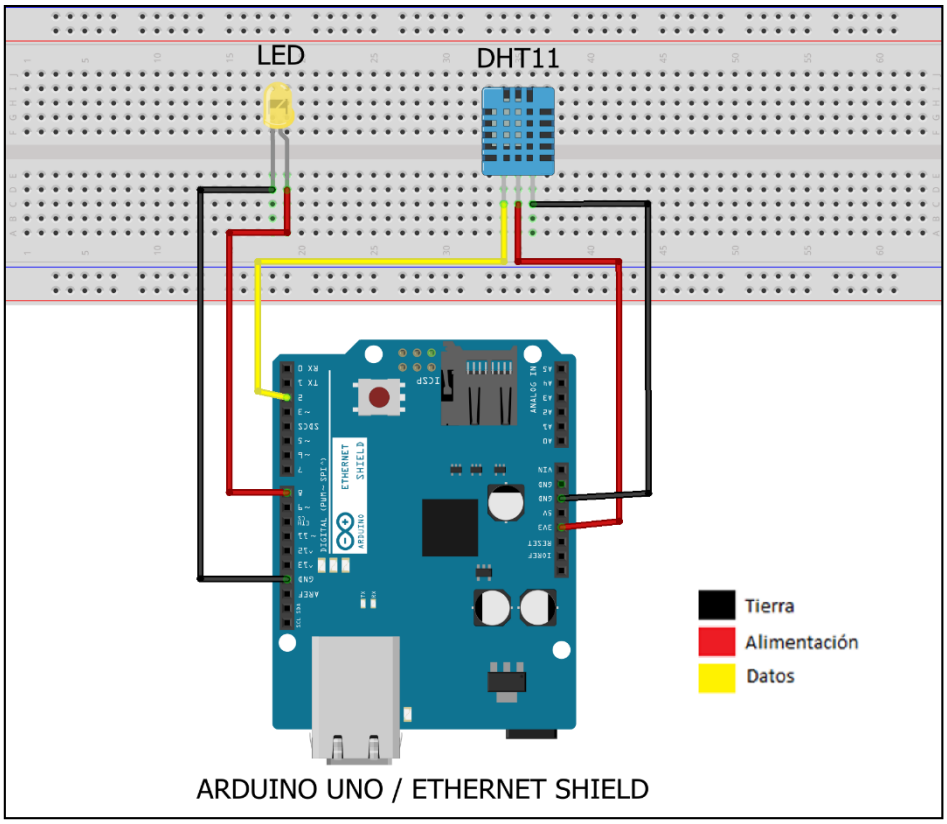

Fig. 6.7 Esquema del circuito realizado con Fritzing

6.4. Arduinoblocks/Arduino IDE

Una vez se ha realizado el esquema del circuito ya se puede proceder a programar el microcontrolador. Existen varias maneras de programar el microcontrolador, ya sea mediante el programa de Arduino IDE que incorpora una lista de todas las placas Arduino e incluso placas que no son marca Arduino pero si utilizan la misma tecnología. Esta programación se realiza con el lenguaje de programación C, y ya viene con un una plantilla de las dos funciones que son necesarias en Arduino: el SETUP() y el LOOP(). En el setup() es donde se inicializan variables y se decide que pines se van a usar, y en cambio, en el loop() va el código destinado a ejecutarse. Una vez finalizado un proyecto se compila y se carga a la Arduino.

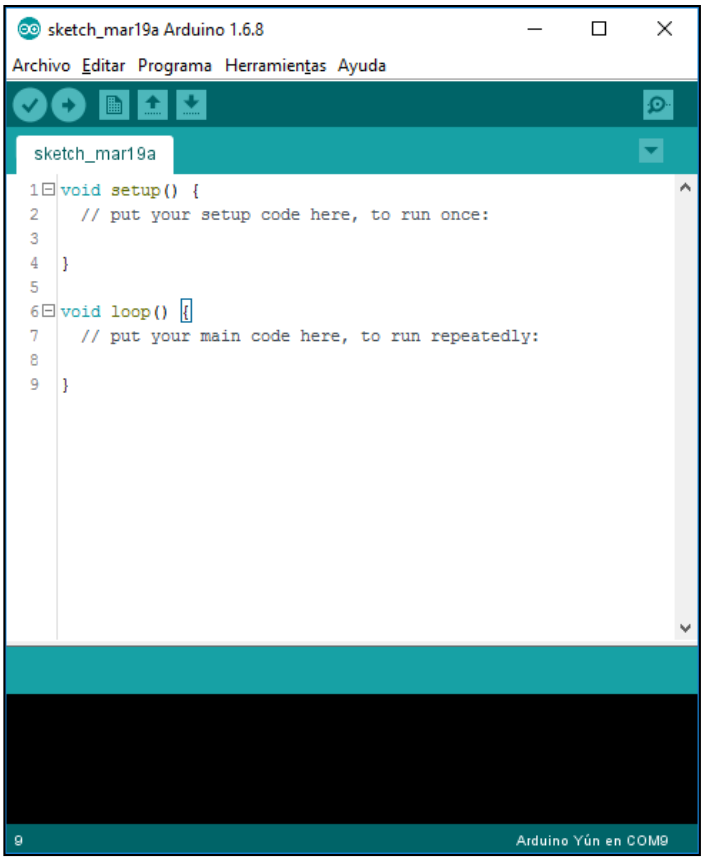

Fig. 6.8 Arduino IDE

Aun así, hoy en día se encuentran programas o páginas web donde la programación de las Arduino se ha intentado simplificar con el auge de las IoT, por lo que se utiliza la programación en bloques. Esto significa que la programación es más sencilla e intuitiva porque solo hay que arrastrar funciones, variables, peticiones al servidor en forma de bloques y rellenar la información de manera sencilla. Internamente, el código sigue siendo C pero cada función se ha traducido a una imagen que internamente contiene el código pertinente.
Un ejemplo de ello es www.arduinoblocks.com , una página web que cuenta con múltiples funciones de programación en que sólo hay que registrarse y empezar un proyecto. También se pueden importar proyectos de otros usuarios ya que hay una base de datos de proyectos de usuarios con comentarios y valoraciones.

Para el proyecto se ha escogido esta manera de programar en bloques ya que facilita y agiliza su ejecución. Para ello, se ha necesitado registrarse y empezar un nuevo proyecto tal y como se muestra en la siguiente imagen:

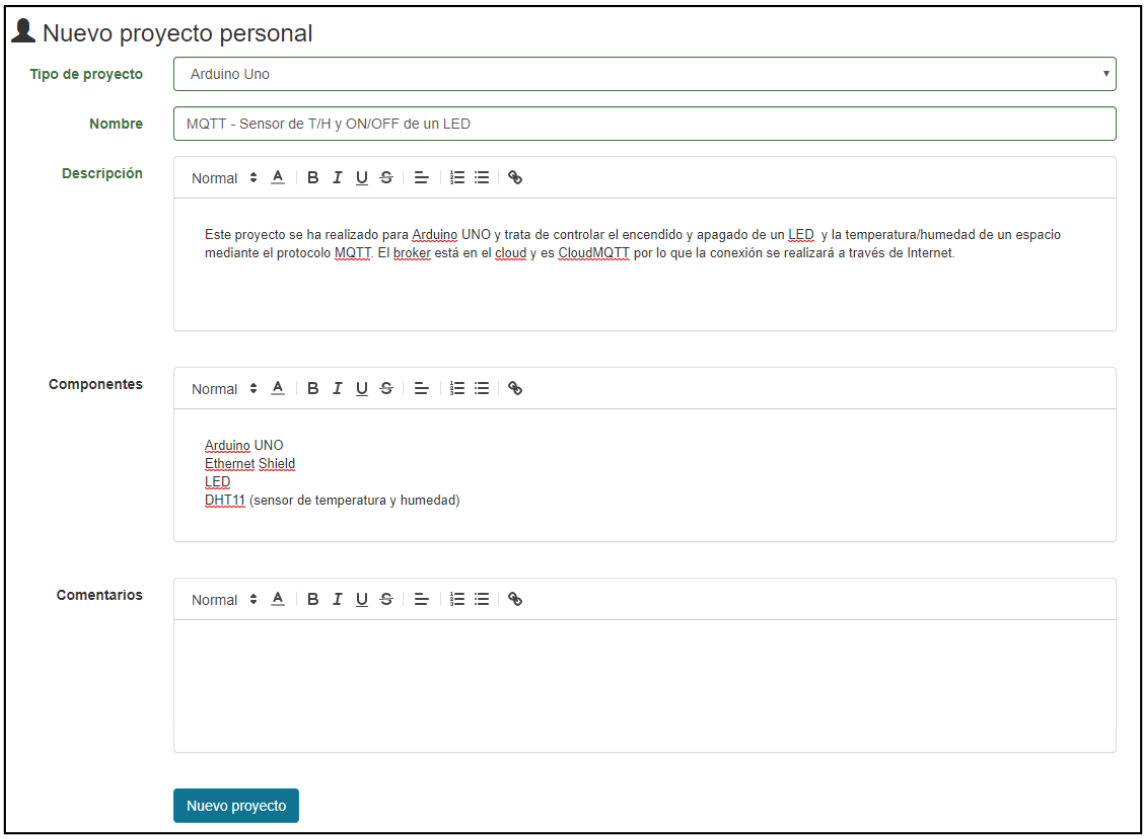

Fig. 6.9 Creación nuevo proyecto en ArduinoBlocks

Una vez se ha iniciado el nuevo proyecto se muestra la página principal que es donde se puede arrastrar cada uno de los elementos de la lista de la izquierda de la imagen al espacio central donde se van uniendo las piezas.

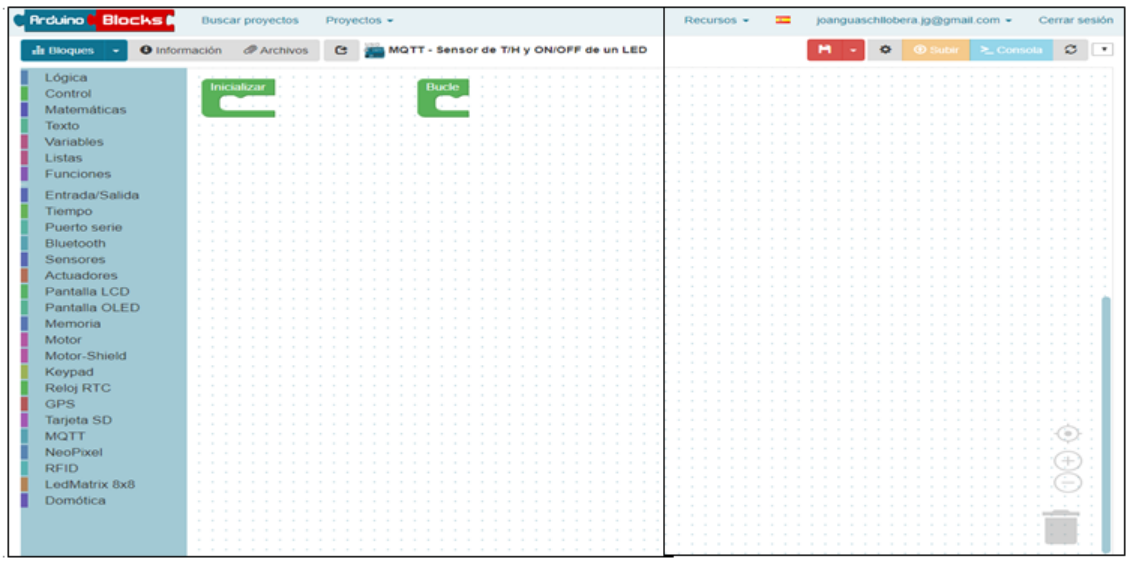

Fig. 6.10 Dashboard ArduinoBlocks

Lo más importante del proyecto es la utilización del protocolo MQTT, pues bien, esta herramienta cuenta con su implementación de manera sencilla. A continuación se muestran las opciones para la petición al servidor MQTT (mediante una conexión ethernet ya que la placa usada es ethernet) y las funciones para publicar temas o suscribirse a topics:

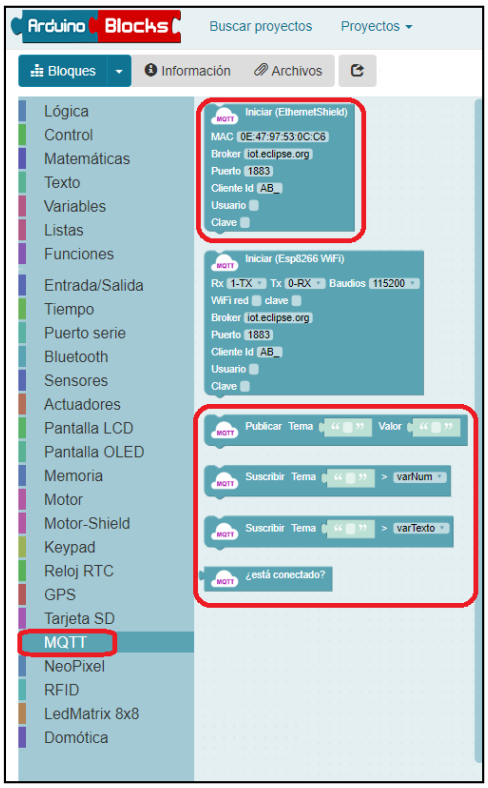

Fig. 6.11 Opciones MQTT de ArduinoBlocks

También se cuenta con varios modelos de sensores, incluido el DHT11. Se muestra en la siguiente imagen:

Fig. 6.12 Tipos de sensores de ArduinoBlocks

Finalmente, después de unir los bloques de las funciones según convenga, el esquema queda de la siguiente manera:

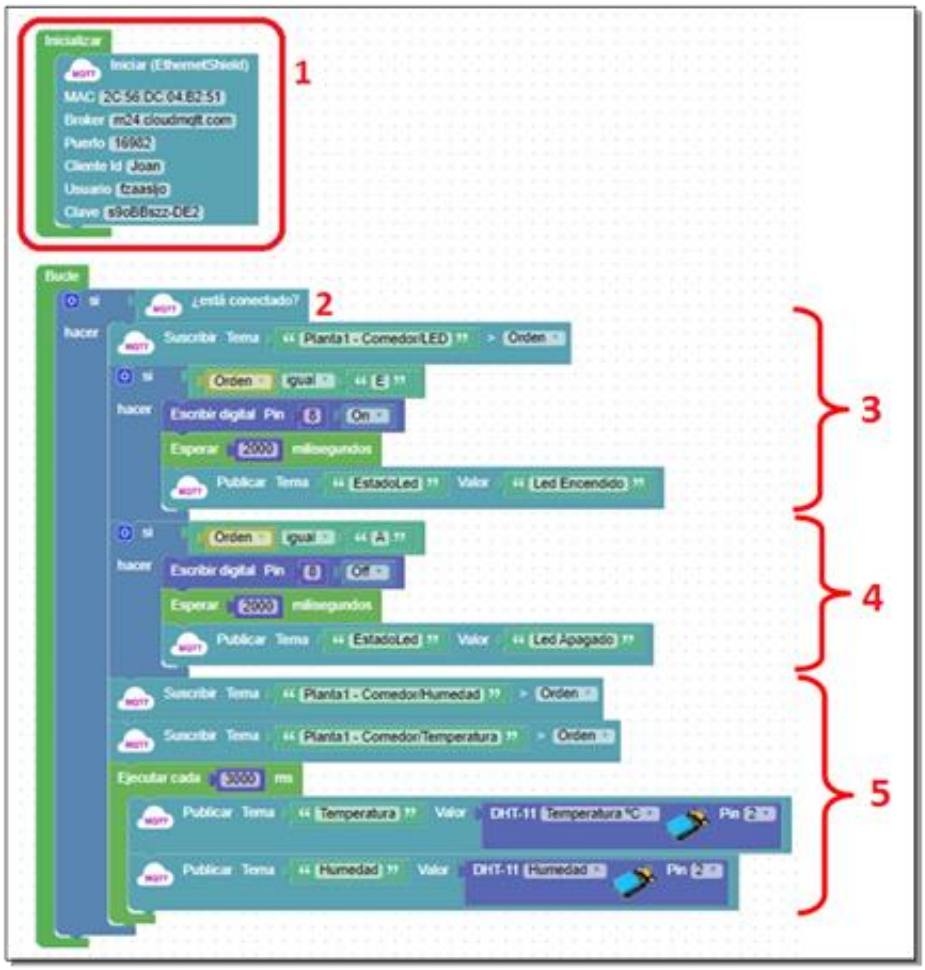

Fig. 6.13 Esquema del diagrama de bloques de ArduinoBlocks

1) En este bloque se inicializa:

- **MAC:** Como la placa ethernet no tiene una MAC asignada, se puede poner cualquiera. En este caso es la dirección 2C:56:DC:04:B2:51.
- **Broker:** Como se usa el servidor de MQTT (CloudMQTT), al registrarse en el servidor web te asignan un servidor gratuito, y en este caso la dirección del servidor es m24.cloudmqtt.com.
- **Puerto:** El servidor MQTT utiliza el puerto 16982, entonces es este el que hay que configurar.
- **Cliente Id:** Este campo solo aporta información sobre quien está usando el servidor pero se puede poner cualquier nombre puesto que no afecta a la configuración.
- **Usuario:** El servidor MQTT ha asignado este usuario y es el que hay que poner en la configuración.
- **Clave:** El servidor MQTT ha asignado esta clave y es la que hay que poner en la configuración.

2) Dentro del bucle lo primero que se ha de hacer es comprobar que realmente se está conectado al servidor.
3) Esta parte de código se centra en el led:
Si se está conectado al servidor MQTT lo primero que hay que hacer es suscribirse a un topic que en este caso el led se suscribe a Planta1 – Comedor/LED. Si el broker envía la orden de encender el led enviando una “E”, Arduino UNO enviará tensión al pin 8 y el led se encenderá. Dos segundos después se publicará un topic sobre el estado del led que aportará información al cliente sobre el estado en que se encuentra el led.
4) Este caso es el mismo que el anterior pero esta vez la orden del broker es una “A”, es decir, apagar el led. Arduino UNO dejará de enviar tensión al pin 8 y el led se apagará. Dos segundos después se publicará el estado del led, en este caso, led apagado.
5) Esta parte de código se centra en el sensor DHT11:
Ahora el sensor se suscribe al topic Planta1–Comedor/Temperatura y Planta1-Comedor/Humedad y cada 5 segundos publicará una lectura de los datos recogidos, es decir, la temperatura y la humedad.

6.4.1. Código

El programa de Arduinoblocks también permite visualizar el código en C por lo que a continuación se muestra:

```
#include <SPI.h>

#include <Ethernet.h>

#include "ABlocksIOTMQTTEthernet.h"

#include "ABlocks_DHT.h"

String s_Orden;

byte mqtt_mac[] = {"2C:56:DC:04:B2:51"};
const char mqtt_broker[]="m24.cloudmqtt.com";
const int mqtt_port=16982;
const char mqtt_user[]="fzaasljo";
const char mqtt_pass[]="s9oBBszz-DE2";
const char mqtt_clientid[]="Joan";
```

```
//ABlocksIOT: ethernet
char mqtt_payload[64];
DHT dht2(2,DHT11);
unsigned long task_time_ms=0;

double mqtt_payload2double(unsigned char *_payload, int _length){
        int i;
        for (i = 0; i<_length && i<64; i++){
                mqtt_payload[i] = _payload[i];
        }
        mqtt_payload[i] = 0;
        return atof(mqtt_payload);
}

String mqtt_payload2string(unsigned char *_payload, int _length){
        int i;
        for (i = 0; i<_length && i<64; i++){
                mqtt_payload[i] = _payload[i];
        }
        mqtt_payload[i] = 0;
        return String(mqtt_payload);
}
void mqtt_callback(char* _topic, unsigned char* _payload, unsigned int _payloadlength){
        double v=mqtt_payload2double(_payload,_payloadlength);
        String vt=mqtt_payload2string(_payload,_payloadlength);
        if(String(_topic)==String(String("Planta1 - Comedor/LED")))s_Orden=vt;
        if(String(_topic)==String(String("Planta1 - Comedor/Humedad")))s_Orden=vt;
        if(String(_topic)==String(String("Planta1 - Comedor/Temperatura")))s_Orden=vt;
}

void mqtt_subscribe(){
        ABlocksIOT.Subscribe(String(String("Planta1 - Comedor/LED")));
        ABlocksIOT.Subscribe(String(String("Planta1 - Comedor/Humedad")));
        ABlocksIOT.Subscribe(String(String("Planta1 - Comedor/Temperatura")));
}

void fnc_dynamic_digitalWrite(int _pin, int _e){
```

```
		pinMode(_pin,OUTPUT);
		digitalWrite(_pin,_e);
}

void setup()
{
		 ABlocksIOT.begin(mqtt_broker,mqtt_port, mqtt_user,mqtt_pass, mqtt_clientid, mqtt_mac, mqtt_callback, mqtt_subscribe);
pinMode(2, INPUT);
dht2.begin();
}



void loop()
{
		ABlocksIOT.loop();
   if (ABlocksIOT.isConnected()) {
     if (String(s_Orden).equals(String("E"))) {
       fnc_dynamic_digitalWrite(8, HIGH);
       delay(2000);
       ABlocksIOT.Publish(String(String("EstadoLed")), String(String("Led Encendido")));

     }
     if (String(s_Orden).equals(String("A"))) {
       fnc_dynamic_digitalWrite(8, LOW);
       delay(2000);
       ABlocksIOT.Publish(String(String("EstadoLed")), String(String("Led Apagado")));

     }
     if((millis()-task_time_ms)>=3000){
		task_time_ms=millis();
       ABlocksIOT.Publish(String(String("Temperatura")), String(dht2.readTemperature()));
       ABlocksIOT.Publish(String(String("Humedad")), String(dht2.readHumidity()));
     }
   }
}
```

6.4.2. Arduinoblocks connector:

Para realizar el traspaso del código a la Arduino UNO desde Arduinoblocks hará falta un programa que hará de enlace para reconocer el puerto USB que está conectado para volcar el código.

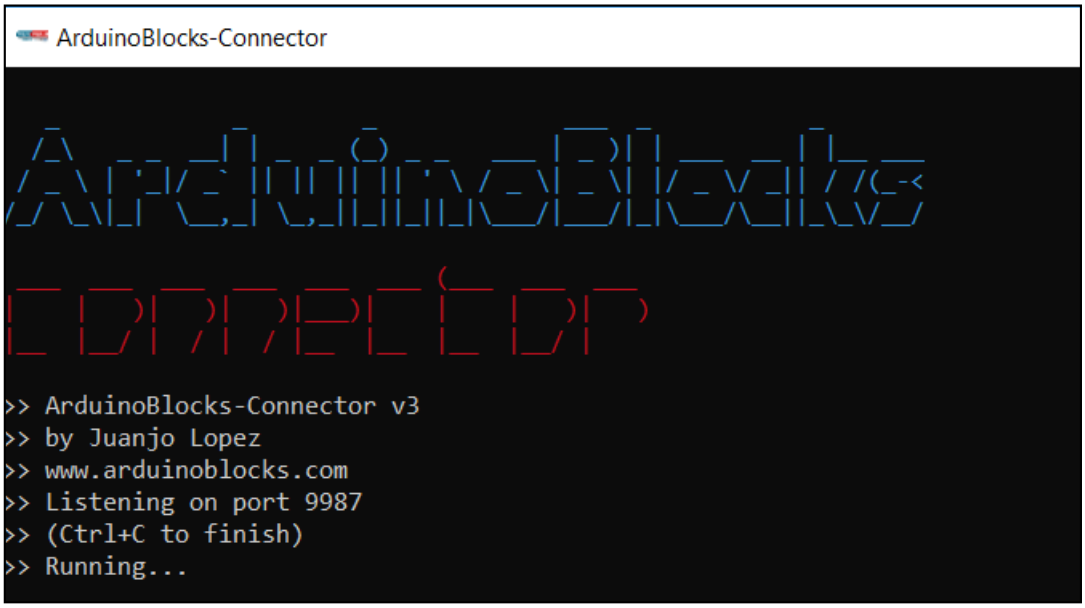

Fig. 6.14 ArduinoBlocksConnector

Finalmente, se transfieren los datos a la Arduino UNO y arduinoblocks avisa al final si se ha realizado con éxito el volcado.

6.5. Configuración broker MQTT

Para la configuración del broker MQTT se ha utilizado el servidor de CloudMQTT, es decir, el servidor está en la nube. Este servidor es configurable de forma sencilla y tiene la opción de tener una cuenta gratuita con 5 accesos simultáneos gratis.

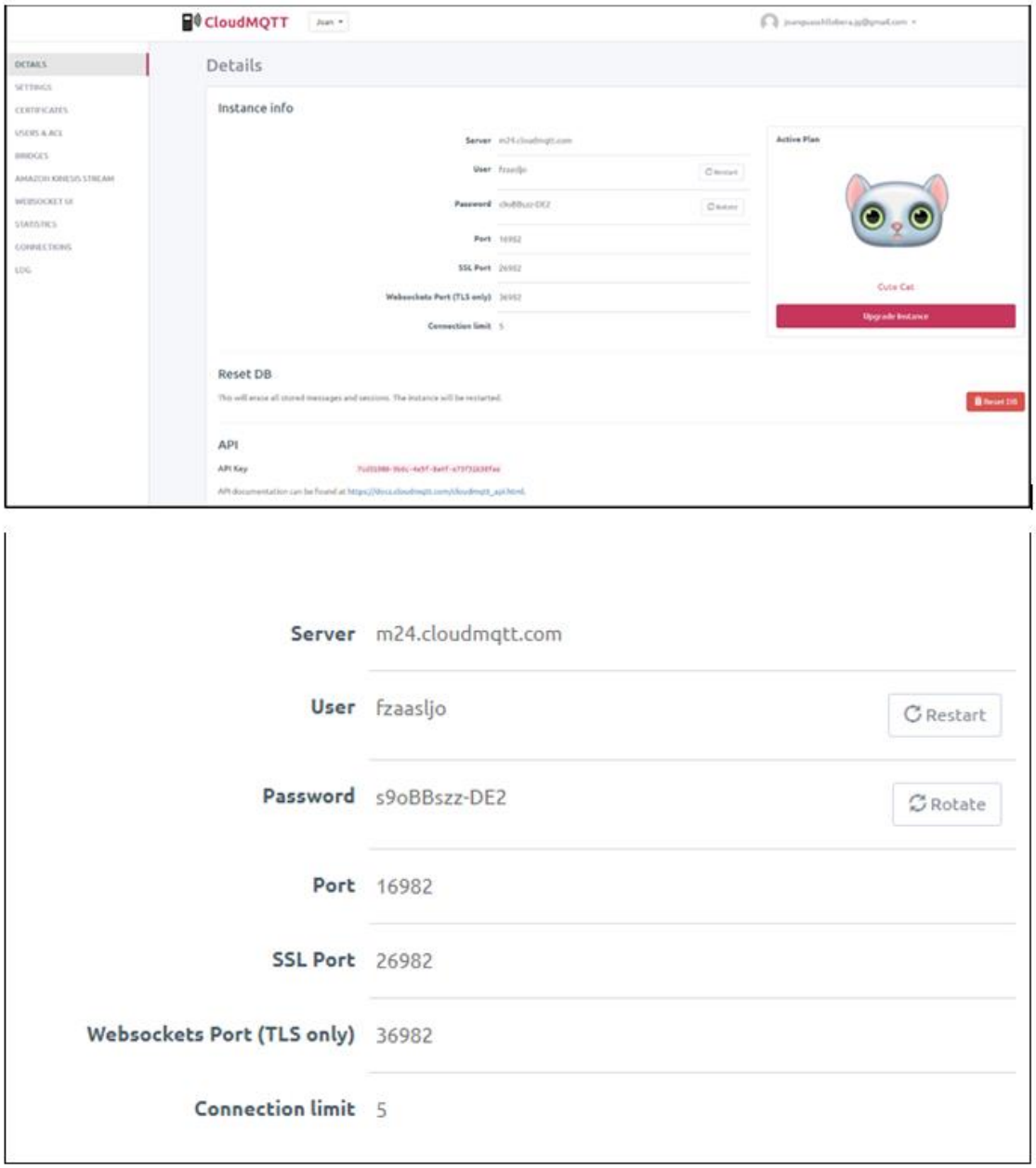

Fig. 6.15 Parámetros de configuración Servidor CloudMQTT

Una vez registrado se te asigna un servidor (m24.cloudmqtt.com), usuario (fzaasljo), contraseña (s9oBBszz-DE2), puerto (16982), puerto SSL (26982), puerto websockets (36982), aunque estos dos últimos no los vamos a necesitar para iniciar la comunicación con el servidor desde arduino.

En el apartado de opciones hay que desactivar la opción de usar el usuario como cliente ID ya que sino solo podrá estar conectado un cliente y en este proyecto se necesita un PC y un móvil como clientes. Se muestra a continuación la imagen:

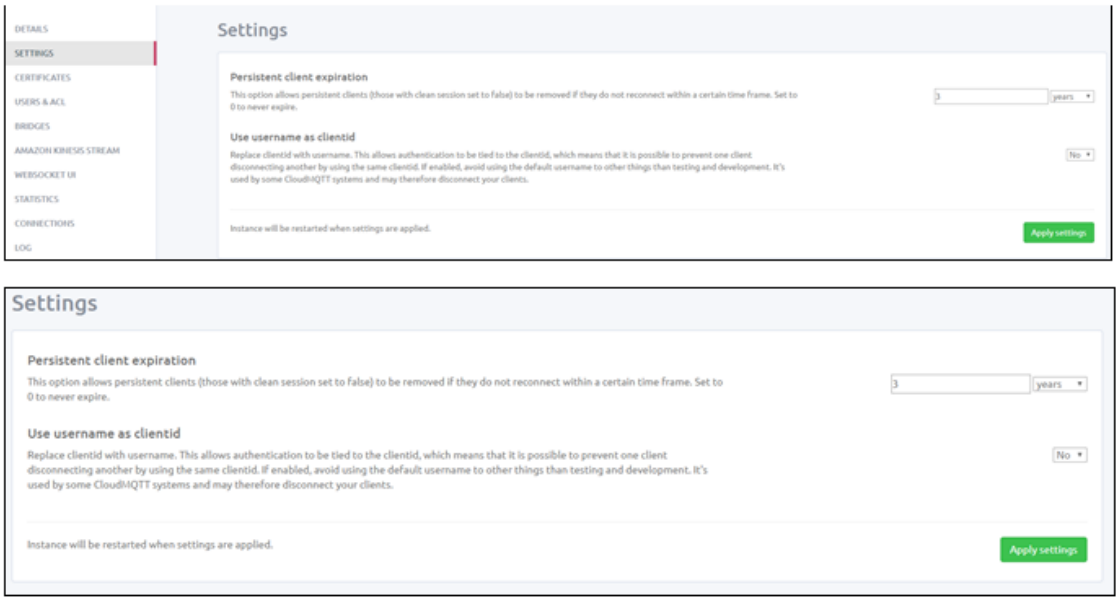

Fig. 6.16 Opciones del servidor CloudMQTT

A continuación hay que configurar los permisos para los usuarios, así que se crea un usuario. Cada usuario tendrá asignadas una serie de reglas llamadas ACL (topic y pattern) y a partir de aquí es lo que se llama crear un topic si se quiere realizar una separación de permisos a la hora de dar órdenes para encender o apagar el led por ejemplo.

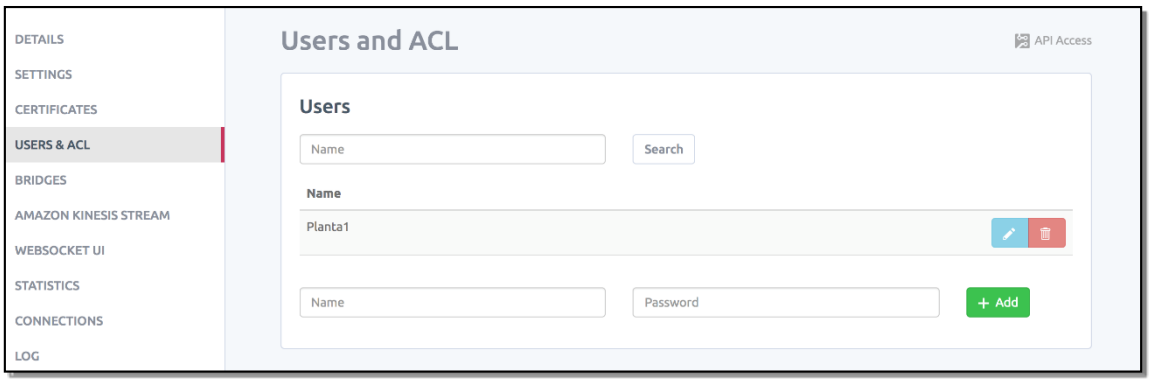

Fig. 6.17 Configuración permisos

Se crea un usuario con una contraseña y a continuación se muestran los topics que se crean en este usuario:

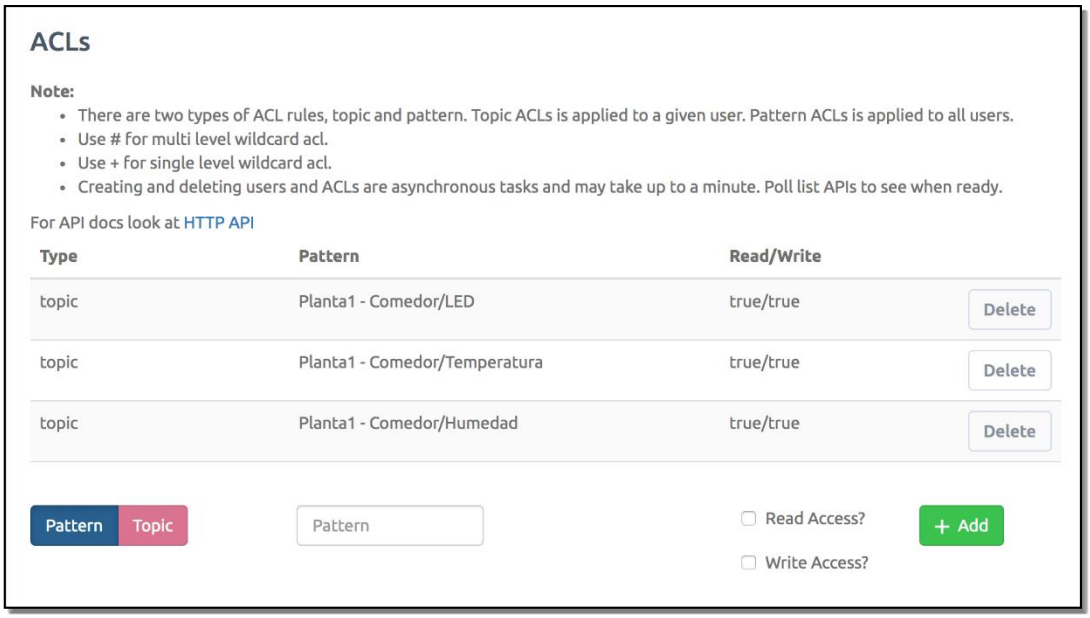

Fig. 6.18 Configuración Topics

Como se observa se han creado 3 topics, para temperatura, humedad y el led. Se les otorga permisos de lectura y escritura para que puedan publicar, es decir, informar del estado del led o dar la información de temperatura y humedad que proviene del DHT11.

6.6. MQTT App Android:

Para el cliente móvil se ha elegido la aplicación IoT MQTT Dashboard, una aplicación que permite la conexión al servidor de CloudMQTT y que permite suscribirse o publicar en topics.
Lo primero es configurar la conexión al servidor de la siguiente manera como se puede ver en la captura:

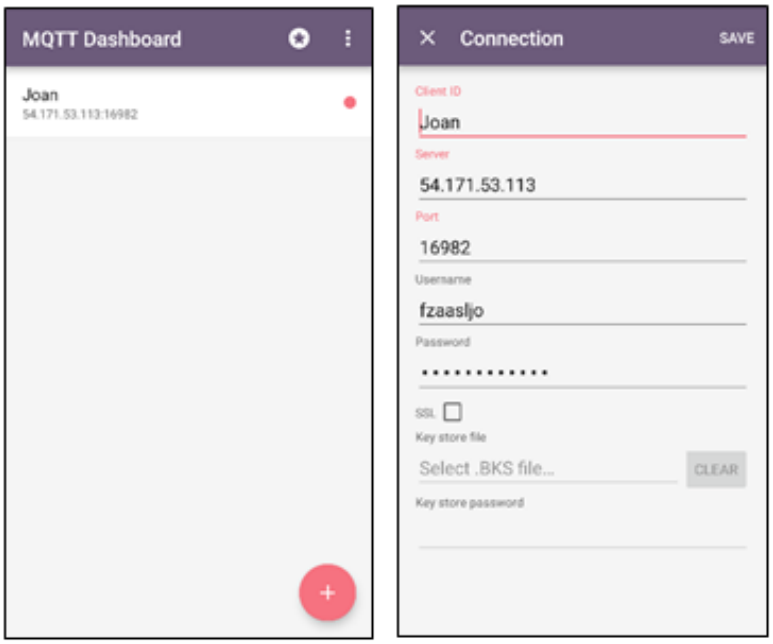

Fig. 6.19 Configuración servidor MQTT Dashboard

Con el botón "+" se accede a la pantalla para rellenar los datos del servidor que se han explicado en el apartado de la configuración del broker MQTT:

- **Client ID:** Joan
- **Servidor:** Aquí se puede poner el nombre del servidor (m24.cloudmqtt.com) o haciendo un ping se puede atacar a la dirección ip (54.171.53.113).

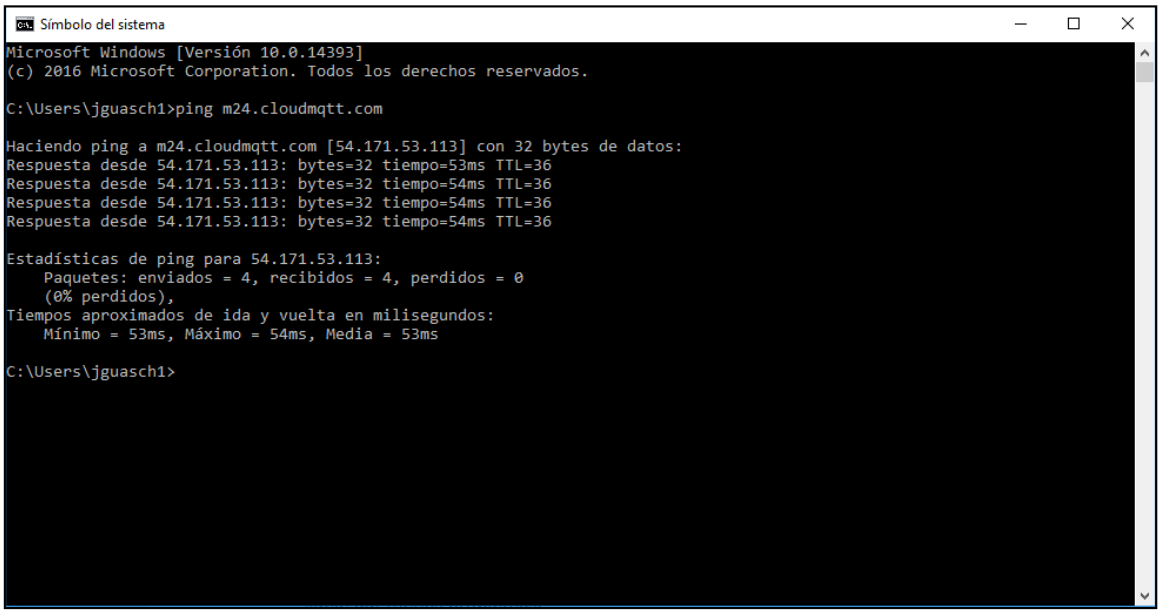

Fig. 6.20 Ping al servidor CloudMQTT

Finalmente se ha introducido la dirección ip ya que suele dar menos problemas.

- **Puerto:** 16982
- **Usuario:** fzaasljo
- **Contraseña:** s9oBBszz-DE2

Una vez rellenada la información se procede a guardarla y a configurar las suscripciones como se muestra en la siguiente imagen:

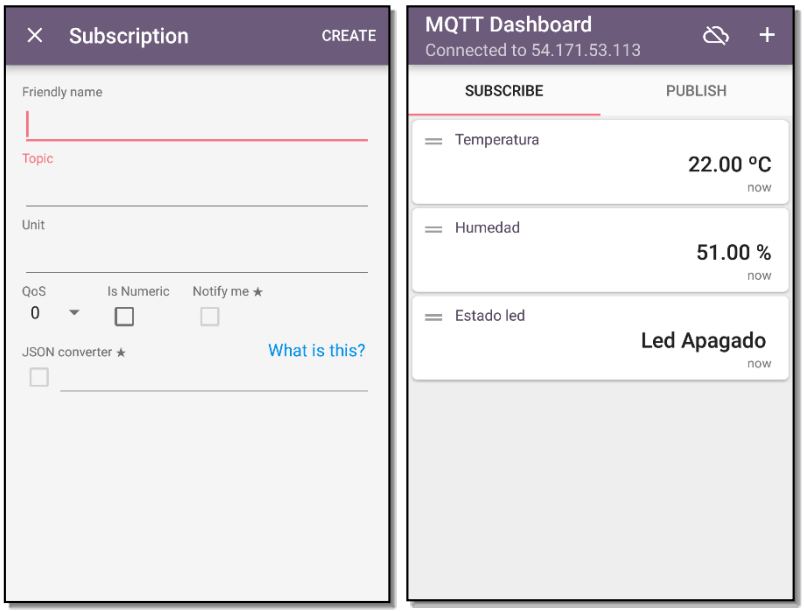

Fig. 6.21 Configuración de las suscripciones

En esta imagen se suscribe a los topics Estado Led, Temperatura y humedad. Para cada uno de ellos habrá que configurar:

- **Nombre descriptivo:** Temperatura (Es un nombre que sirve para entender que es. No necesariamente tiene que ser el mismo que el nombre del topic).
- **Topic:** T (Es el nombre del topic configurado en el broker MQTT).
- **Unidades:** ºC (Son las unidades en que se interpretará el valor resultante).
- **QoS:** 0 (La aplicación ofrece la posibilidad de establecer que grado de QoS se quiere entre QoS 0, QoS 1 y QoS 2. En este caso como lo que se publica no es crítico no hace falta recibir un ACK de confirmación).

Una vez rellenada la información se crean la suscripciones y se procede a configurar las publicaciones como se muestra en la siguiente imagen:

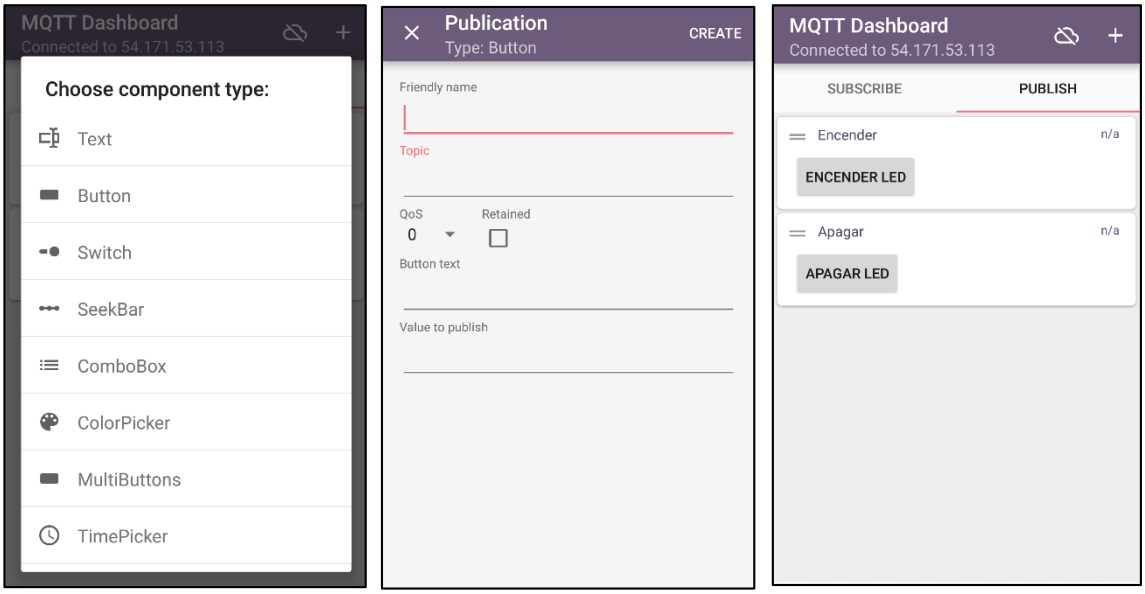

Fig. 6.22 Configuración de las publicaciones

En esta imagen se publica al topic Estado Led y la idea será publicar información para encender y para apagar el led. Para ello habrá que crear dos botones, uno para encender y otro par apagar el led y en cada uno habrá que configurar:

- **Nombre descriptivo:** Encender en el primer botón y apagar en el otro (Es un nombre que sirve para entender que es. No necesariamente tiene que ser el mismo que el nombre del topic).
- **Topic:** Estado Led (Es el nombre del topic configurado en el broker MQTT).
- **QoS:** 0 (como se ha comentado en el apartado de configuración de las suscripciones, se utilizará el QoS 0 por el mismo motivo).
- **Texto del botón:** Encender en el primer botón y apagar en el otro. (Es el texto que hay en el botón).
- **Valor que se quiere publicar:** “E” para encender y “A” para apagar. En el broker MQTT y en el microcontrolador de Arduino UNO se han configurado estos valores por lo que para realizar las acciones que se requieren se deberá usar la misma nomenclatura.

6.7. Test

Finalmente, como la configuración en Arduino UNO, el broker MQTT y la aplicación Android de MQTT está realizada sólo queda hacer la prueba de que este montaje funciona.

Habrá que conectar la Arduino UNO al router mediante un cable ethernet, comprobar que el servidor está funcionando y que se tiene conexión al servidor desde el PC y el móvil.

Cuando el servidor se enciende lo que se ve en el cliente del PC es lo siguiente:

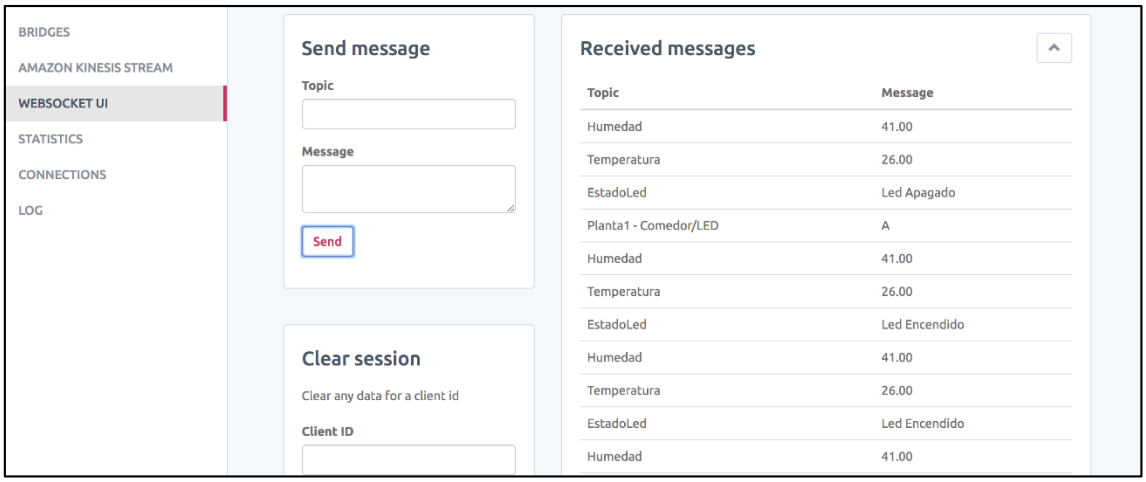

Fig. 6.23 Información de los sensores

Cada 2 segundos se recibe la información del estado del LED y cada 3 segundos se recibe la información del sensor de temperatura y humedad.

Para publicar enviar un mensaje para modificar el estado del LED hay que enviar un mensaje. Para ello hay que hacer referencia al topic al cuál va dirigido y escribir el mensaje. En la siguiente imagen se puede apreciar esta explicación:

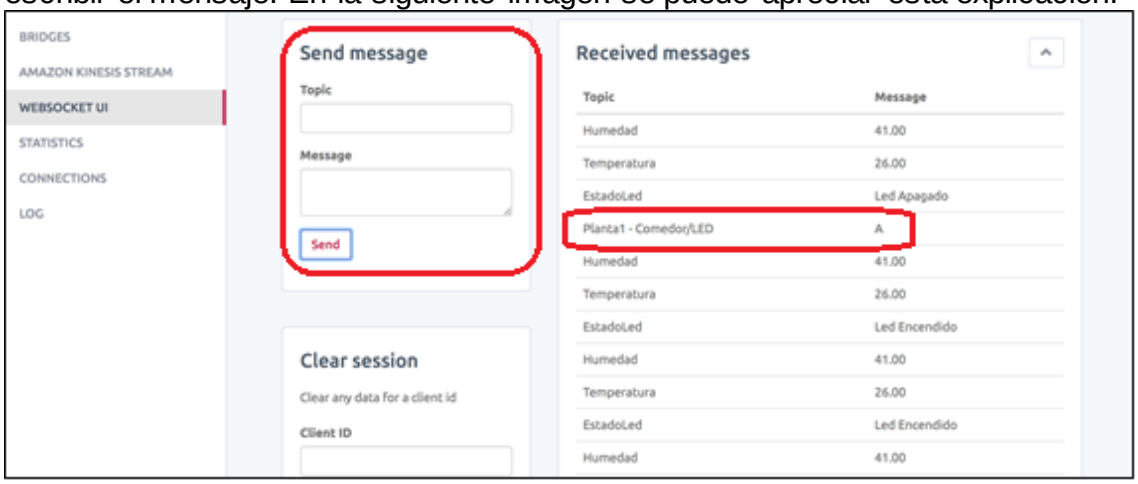

Fig. 6.24 Publicación del estado del LED

Lo que se pretende es apagar el LED ya que el estado que tenía era “Led Encendido” por lo que se escribe al topic “Planta1-Comedor/LED” y se manda el mensaje de apagar con la letra “A”. Al cabo de 2 segundos se recibe la información de que se ha apagado el led y se lee “Led Apagado”.

Y cuando se accede desde el cliente móvil lo que se ve es lo siguiente:

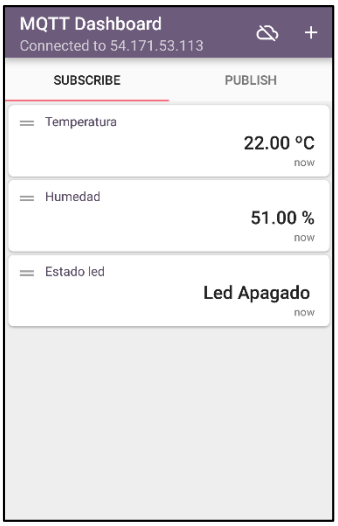

Fig. 6.25 Pantalla principal MQTT Dashboard

En la pantalla principal es donde se ve la información de los sensores y si se quiere cambiar el estado del led habrá que publicar un mensaje para apagar el led. En la siguiente imagen se ve como se puede encender o apagar el led pulsando los botones de encendido y apagado ya que internamente se está publicando al topic Planta1-Comedor/LED y se está mandando un mensaje de encendido “E” o apgado “A”.

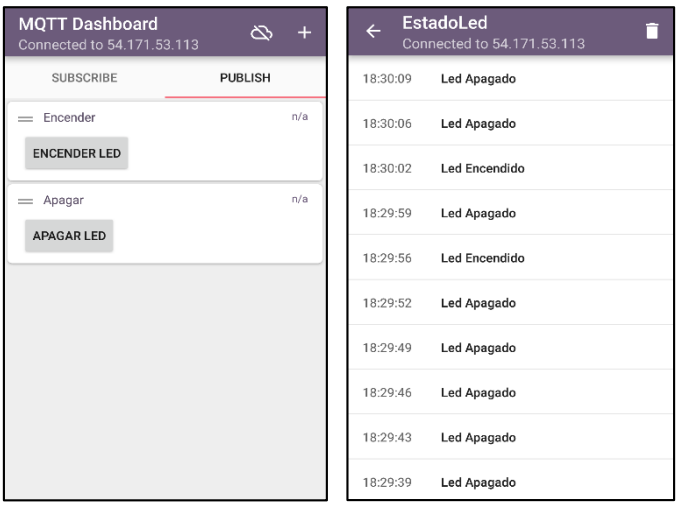

Fig. 6.26 Publicación del estado del LED

CAPÍTULO 7. VALORACIONES Y CONCLUSIONES

En este capítulo, el último del proyecto, se comentan las conclusiones del proyecto, tanto las generales como las personales.

7.1. Conclusiones generales

La internet de las cosas se ha establecido de manera fija en la sociedad, que cada vez depende más de la tecnología, por tanto, se intenta actualizar o inventar nuevos protocolos de comunicación para poder satisfacer las demandas de los usuarios.

Se prevé que cada año aumenten los dispositivos conectados a internet. Estos dispositivos inteligentes (sensores, móviles, drones, coches, luces, etc) generan un gran tráfico de datos en la red. Según la empresa Cisco, se prevé que en el año 2022 los dispositivos conectados superarán el 50% del total de conexiones.

A continuación, se muestra la gráfica del estudio comentado con anterioridad.

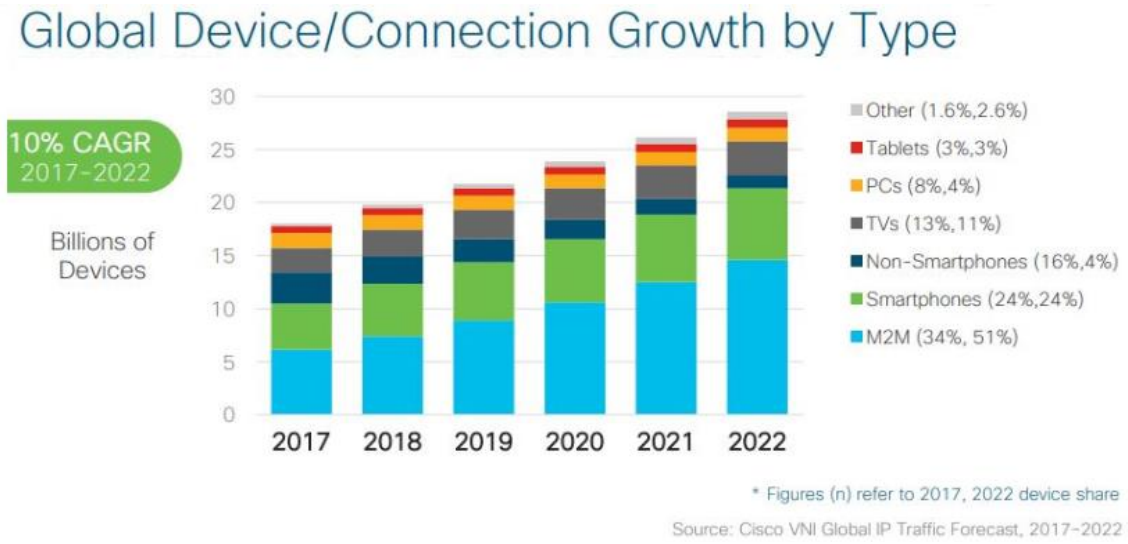

Fig. 7.1 Estudio CISCO crecimiento de dispositivos conectados (ver [12])

El grosor del tráfico, como se ha comentado anteriormente, lo generan los dispositivos M2M, donde se prevé que la mayor parte sean de dispositivos conectados en casa. La misma empresa, CISCO, prevé que sea un total del 50% del tráfico total generado por estos dispositivos.

A continuación, se puede observar el estudio realizado por CISCO.

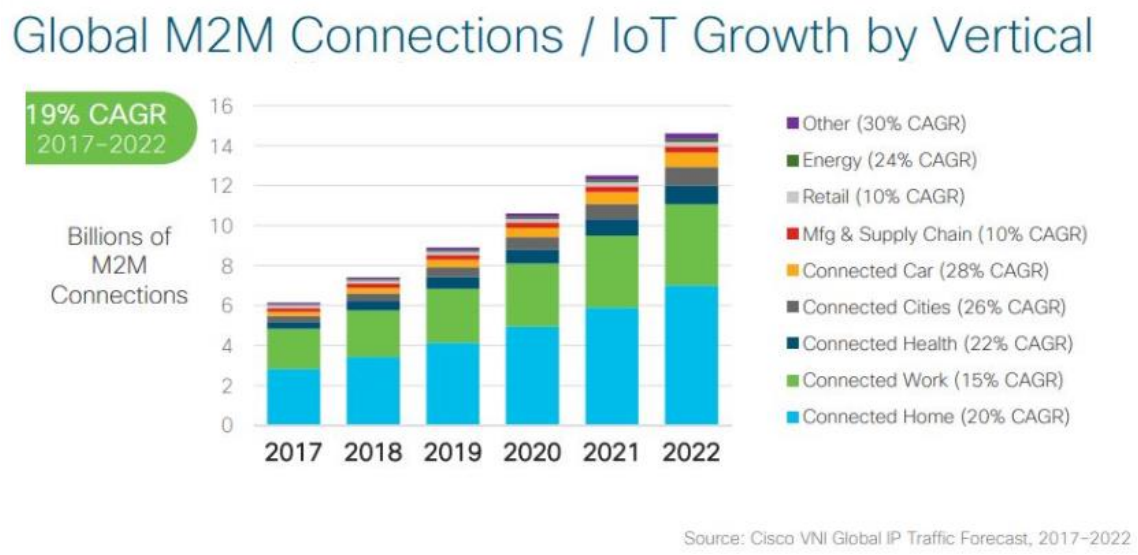

Fig. 7.2 Estudio CISCO de M2M conectados (ver [12])

Como se observa en los dos estudios, los dispositivos más importantes y que generan una gran cantidad de datos son los dispositivos M2M conectados en las casas. Estos datos tienen que ser procesados por los diferentes protocolos y estándares de comunicación. Y como se ha mostrado en este proyecto, uno de los más importantes es el MQTT.

El estándar de comunicación MQTT ayuda a resolver el problema del ancho de banda que generan los dispositivos conectados a la red. A causa de que necesita muy poco ancho de banda para facilitar la comunicación entre dispositivos M2M (leds, diferentes tipos de sensores, etc).

Por otro lado, al tener tantos dispositivos conectados a la red y generando constantemente un flujo de datos, es muy importante solucionar el problema de seguridad, factor que se puede solucionar si se desea con este protocolo.

7.2. Conclusiones personales

La realización de este proyecto, principalmente, nos ha aportado conocimientos en el ámbito de los protocolos y estándares de comunicación usados en las IoT. Cuando se empezó el proyecto solo teníamos un conocimiento básico de las aplicaciones útiles y algún que otro protocolo de comunicación, como por ejemplo, HTTP.

Ha sido interesante estudiar con más profundidad los protocolos más usados: HTTP, CoAP, WebSocket, XMPP y MQTT, ya que gracias a esto hemos podido entender cómo se solucionan los problemas que genera vivir en un mundo tecnológico, donde se genera un gran tráfico de datos de manera constante.

Gracias a profundizar en el protocolo MQTT durante el proyecto, ahora entendemos y somos capaces de realizar una instalación de IoT óptima en, por ejemplo, nuestras respectivas viviendas o de amigos y familiares.

BIBLIOGRAFÍA

[1] IoT
Rose K., Eldridge S., Chapin L., *La internet de las cosas*, Internet Society, Octubre 2015

[2] ARDUINO UNO
https://www.arduino.cc/

[3] ESTRUCTURA PROGRAMACIÓN MQTT
https://docsmedialab.readthedocs.io/es/latest/quickReference/mqtt.html

[4] TIPOS DE SENSORES ARDUINO
https://aprendiendoarduino.wordpress.com/category/sensores/

[5] CARACTERÍSTICAS ARDUINO UNO
http://www.iescamp.es/miarduino/2016/01/21/placa-arduino-uno/

[6] ARQUITECTURA Y ESTRUCTURA DE UN MENSAJE MQTT
https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/

[7] DEFINICIÓN MQTT
https://www.ibm.com/developerworks/ssa/library/IoT-mqtt-why-good-for-IoT/index.html

[8] PROTOCOLOS DE COMUNICACIÓN
https://www.arrow.com/es-mx/research-and-events/articles/protocols-for-the-internet-of-things

[9] XMPP
https://hipertextual.com/archivo/2014/07/protocolo-xmpp/

[10] QoS MQTT
https://mntolia.com/mqtt-qos-levels-explained/

[11] PLACAS CON HARDWARE LIBRE
https://blogthinkbig.com/4-alternativas-arduino-beaglebone-raspberrypi-nanode-waspmote

[12] IoT DISPOSITIVOS CONECTADOS
https://www.interxion.com/es/blogs/2018/12/avalancha-de-datos-iot-inundara-las-redes-y-los-centros-de-datos/